ICON 2020

17th International Conference on Natural Language Processing

Proceedings of the Conference

December 18 - 21, 2020
Indian Institute of Technology Patna, India

Preface

Research in Natural Language Processing (NLP) has taken a noticeable leap in recent years. The tremendous growth of information on the web and its easy access has stimulated a large interest in the field. India, with multiple languages and continuous growth of Indian language content on the web, makes a fertile ground for NLP research. Moreover, the industry is keenly interested in obtaining NLP technology for mass use. Internet search companies are increasingly aware of the large market for processing languages other than English. For example, search capability is needed for content in Indian and other languages. There is also a need for searching content in multiple languages, and making the retrieved documents available in the language of the user. As a result, a strong need is being felt for machine translation to handle this large instantaneous use. Information Extraction, Question Answering Systems, and Sentiment Analysis are also showing up as other business opportunities.

These needs have resulted in two welcome trends. First, there is a much wider student interest in getting into NLP at both postgraduate and undergraduate levels. Many students interested in computing technology are getting interested in natural language technology, and those interested in pursuing computing research are joining NLP research. Second, the research community in academic institutions and government funding agencies in India have joined hands to launch consortia projects to develop NLP products. Each consortium project is a multi-institutional endeavour working with a common software framework, common language standards, and common technology engines for all the different languages covered in the consortium. As a result, it has already led to the development of basic tools for multiple languages that are interoperable for machine translation, cross-lingual search, handwriting recognition, and OCR.

In this backdrop of increased student interest, greater funding, and most importantly, common standards and interoperable tools, there has been a spurt in research in NLP on Indian languages whose effects we have just begun to see. A great number of submissions reflecting good research is a heartening matter. There is an increasing realization to take advantage of features common to Indian languages in machine learning. It is a delight to see that such features are not just specific to Indian languages but to a large number of languages of the world, hitherto ignored. The insights so gained are furthering our linguistic understanding and will help in technology development for hopefully all languages of the world.

For machine learning and other purposes, linguistically annotated corpora using the common standards have become available for multiple Indian languages. They have been used for the development of basic technologies for several languages. A larger set of corpora are expected to be prepared in the near future.

These conference proceedings contain papers selected for presentation in technical sessions of ICON-2020. We are thankful to our excellent team of reviewers from all over the globe who deserve full credit for the hard work of reviewing the high-quality submissions with rich technical content. From 130 submissions, 66 papers were selected, 29 long papers, 34 short papers, 3 doctoral consortium papers, representing a variety of new and interesting developments, covering a wide spectrum of NLP areas and core linguistics. Besides presentations, the conference also hosted 2 tutorials, 1 workshop, 3 shared tasks, and 18 system demonstrations.

We are deeply grateful to Prof. David Yarowsky from John Hopkins University (USA), Prof. Iryna Gurevych from Technische Universität Darmstadt (Germany), and Prof. Eduard Hovy from Carnegie Mellon University for giving the keynote lectures at ICON. We also extend our heartfelt thanks to Dr Soujanya Poria, Singapore University of Technology and Design, Singapore for giving the invited talk at

ICON.

We thank all the area chairs for the various tracks at ICON 2020, especially, Sobha Lalitha Devi (Language Resources, NLP Language Documentation and Preservation), Ashwini Vaidya, Pawan Goyal (Syntax and Lexical Semantics), Praveen Kumar G S (Named Entity Recognition, Question Answering, Information Extraction, Dialogue Systems), Amitava Das, Radhika Mamidi (Sentiment and Emotion Analysis), Karunesh Arora, Sandipan Dandapat (Machine Translation), Vasudeva Varma, Dipankar Das (Summarization, Natural Language Generation, Information Retrieval and Text Mining), C V Jawahar (Multimodality, Speech Recognition, Speech Synthesis), Raksha Sharma, Nikesh Garera, (NLP for Digital Humanities, NLP for Education), Samar Husain (Ethics in NLP, Cognitive Modelling and Psycholinguistics), Karthik Sankaranarayanan, Ashutosh Modi (Machine Learning Applications to NLP, Interpretability and Explainability of NLP models). We also thank Gurpreet Singh Lehal, Sanjay Dwivedi, Rajeev R R, Sanjeev Gupta, Neeraj Mogla, Amba Kulkarni (Co-Chairs, Tools Contest), Sudip Kumar Naskar, Sriparna Saha (Co-Chairs, Workshop/Tutorial), Preethi Jyothi (Doctoral Consortium Chair) for taking the responsibilities of the events.

We are thankful to the team members of the Artificial Intelligence-Natural Language Processing-Machine Learning (AI-NLP-ML) Group of the Department of Computer Science and Engineering for making the organization of the event at the Indian Institute of Technology Patna (IIT Patna) a success. We heartily express our gratitude to Pushpak Bhattacharyya, Asif Ekbal, Sriparna Saha, Soumitra Ghosh, Ratnesh Joshi, Prabhat Kumar Bharti, Gitanjali Singh, Tirthankar Ghosal, Apoorva Singh, and other AI-NLP-ML team members at IIT Patna for their timely help with sincere dedication to make this conference a success. We also thank and all those who came forward to help us with this task.

Finally, we thank all the researchers who responded to our call for papers and all the participants of ICON-2020, without whose overwhelming response the conference would not have been a success. We wholeheartedly thank all the reviewers who accepted our invitation and spent their valuable time reviewing the papers to maintain their high international standards. We thank the session chairs for finding out time for our conference.

December 2020
Patna
.

Pushpak Bhattacharyya-PC Co-chair
Dipti Misra Sharma-PC Co-chair
Rajeev Sangal-General Chair

Asif Ekbal-Organizing Committee Chair

Conference General Chair

Rajeev Sangal, IIIT Hyderabad, India

Program Chairs:

Dipti Misra Sharma, IIIT Hyderabad, India (Co-Chair)
Pushpak Bhattacharyya, IIT Bombay, India (Co-Chair)

Organizing Chair:

Asif Ekbal, IIT Patna, India

Program Committee:

Sobha Lalitha Devi, AU-KBC Research Centre, Anna University
Ashwini Vaidya, IIT Delhi
Pawan Goyal, IIT Kharagpur, India
Praveen Kumar G S, Samsung
Amitava Das, Wipro AI Lab
Radhika Mamidi, IIIT Hyderabad, India
Karunesh Arora, CDAC
Sandipan Dandapat, Microsoft
Vasudeva Varma, IIIT Hyderabad
Pushpak Bhattacharyya, IIT Bombay, India
Dipti Misra Sharma, IIIT Hyderabad
Dipankar Das, Jadavpur University
C V Jawahar, IIIT Hyderabad
Raksha Sharma, IIT Roorkee
Samar Husain, Indian Institute of Technology Delhi, India
Nikesh Garera, Flipkart
Karthik Sankaranarayanan, IBM Reseach
Ashutosh Modi, Indian Institute of Technology Kanpur
Tanmoy Charaborty, IIIT Delhi
Sriparna Saha, IIT Patna, India
Anil Kumar Vuppala, IIIT Hyderabad
Aditya Joshi, CSIRO
Girish Palshikar, Tata Consultancy Services Limited
Manish Srivastava, IIIT Hyderabad
Vishal Goyal, Punjabi University, Patiala
Sudip Kumar Naskar, Jadavpur University
Sudeshna Sarkar, IIT Kharagpur, India
Anoop Kunchukuttan, Microsoft AI and Research
Shad Akhtar, IIIT Delhi

Tools Contest Chairs:

Gurpreet Singh Lehal, Punjabi University Patiala
Sanjay Dwivedi, Central University, Lucknow
Rajeev R R, ICFOSS, Trivandrum
Sanjeev Gupta, Google, Bangalore
Neeraj Mogla, Flipkart, USA
Amba Kulkarni, University of Hyderabad

Workshop/Tutorial Chairs:

Sudip Kumar Naskar, Jadavpur University
Sriparna Saha, IIT Patna, India

Doctoral Consortium Chairs:

Preethi Jyothi, IIT Bombay, India

Invited Speakers:

Prof. David Yarowsky, John Hopkins University, USA
Prof. Iryna Gurevych, Technische Universität Darmstadt, Germany
Prof. Eduard Hovy, Carnegie Mellon University
Dr. Soujanya Poria, Singapore University of Technology and Design, Singapore

Referees

We gratefully acknowledge the excellent quality of refereeing we received from the reviewers. We thank them all for being precise and fair in their assessment and for reviewing the papers in time.

Muhammad Abulaish, South Asian University
Sriparna Saha, IIT Patna, India
Zishan Ahmad, IIT Patna, India
Md. Shad Akhtar, Indraprastha Institute of Information Technology, Delhi
Ashish Anand, Indian Institute of Technology Guwahati, India
C Anantaram, Indraprastha Institute of Information Technology, Delhi
Mohd Zeeshan Ansari, Jamia Millia Islamia
Karunesh Arora, CDAC
Rakesh Balabantaray, IIIT Bhubaneswar
Kalika Bali, Microsoft Research Labs
Somnath Banerjee, University of Milano-Bicocca
Srinivas Bangalore, Interactions Corp
Debajyoty Banik, IIT Patna, India
Biswan Barik, GE Global Research
Kingshuk Basak, Samsung Research and Development
Brendan Bena, Drury University
Riyaz A. Bhat, Interactions LLC
Mehar Bhatia, IIIT-Delhi, India
Michael Carl, Kent State University
Tanmoy Chakraborty, Indraprastha Institute of Information Technology Delhi (IIIT-D), India
Sanjay Chatterji, Indian Institute Of Information Technology Kalyani
Dushyant Singh Chauhan, IIT Patna, India
Kushal Chawla, University of Southern California
Soumya Chennabasavaraj, Flipkart
Manoj Chinnakotla, Microsoft
Monojit Choudhury, Microsoft Research
Thomas Conley, University of Colorado, Colorado Springs
Sandipan Dandapat, Microsoft
Dipankar Das, Jadavpur University
Amitava Das, Wipro AI Lab
Kumar Gourav Das, Jadavpur University
Ayan Das, IIT Kharagpur, India
Niladri Sekhar Dash, Linguistic Research Unit, Indian Statistical Institute, Kolkata
Arkadia De, Indian Institute of Technology Hyderabad, India
Alok Debnath, International Institute of Information Technology, Hyderabad, India
Kuntal Dey, Accenture Technology Labs
Gihan Dias, University of Moratuwa
Sri Harsha Dumpala, Vector Institute, and Dalhousie University
Pratik Dutta, IIT Patna, India
Hridoy Sankar Dutta, IIIT Delhi, India
Indranil Dutta, Jadavpur University
Mohd Fazil, Madanapalle Institute of Technology & Science, Madanapalle, Chittoor, India
Mauajama Firdaus, IIT Patna, India

Suryakanth V Gangashetty, KLEF Deemed to be University, Green Fields
Nikesh Garera, Flipkart
Ibrahim Gashaw, Mangalore University
Tirthankar Ghosal, IIT Patna, India
Deepanway Ghosal, Singapore University of Technology and Design
Soumitra Ghosh, IIT Patna, India
Prabhat Kumar Bharti, IIT Patna, India
Sanjukta Ghosh, IIT BHU, India
Souvick Ghosh, San Jose State University
Pranav Goel, Wadhwani Institute for Artificial Intelligence
Vishal Goyal, Punjabi University Patiala
Pawan Goyal, IIT Kharagpur, India
Praveen Kumar GS, Samsung
Kamal Kumar Gupta, IIT Patna, India
Deepak Gupta, IIT Patna, India
Harald Hammarstrom, Uppsala University
Rejwanul Haque, Dublin City University
Mohammed Hasanuzzaman, Cork Institute of Technology, Dublin, Ireland
Samar Husain, Indian Institute of Technology Delhi, India
Nikhil Jaiswal, TCS Research
C V Jawahar, IIIT Hyderabad
Girish Jha, Jawaharlal Nehru University
Saurav Jha, University of Lorraine
Harimohan Jha, IIT Kharagpur
Aditya Joshi, CSIRO
Nilesh Joshi, IIT Bombay, India
Preethi Jyothi, IIT Bombay, India
Mitesh M. Khapra, IIT Madras, India
Sunil Kumar Kopparapu, TCS Research, and Innovation, Mumbai
Alapan Kuila, IIT Kharagpur, India
Amba Kulkarni, University of Hyderabad,
Malhar Kulkarni, IIT Bombay, India
Pranaw Kumar, CDAC Mumbai
Niraj Kumar, Senior Researcher, and Manager, Samsung Research Institute India, Bangalore
Ritesh Kumar Dept. of Linguistics, Dr. Bhimrao Ambedkar University, Agra
Abhinav Kumar, NIT Patna, India
Abhishek Kumar, IIT Patna, India
Vijay Kumar, Ministry of Electronics and Information Technology
Anil Kumar Singh, IIT BHU Varanasi, India
Divya Kumari, IIT Patna, India
Rina Kumari, IIT Patna, India
Anoop Kunchukuttan, Microsoft AI and Research
Bibekananda Kundu, Centre for Development of Advanced Computing (CDAC) Kolkata
Sobha Lalitha Devi, AU-KBC Research Centre, Anna University
Gurpreet Lehal, Punjabi University
Abhijith Madan, International Institute of Information Technology, Bangalore, India
Avinash Madasu, Samsung R & D Institute Bangalore
Sainik Mahata, Jadavpur University

Abhra Majumdar, IIT Kharagpur, India
Shrikant Malviya, IIIT Allahabad, India
Radhika Mamidi, IIIT Hyderabad, India
Soumik Mandal, Rutgers University
Soumil Mandal, SRM University
Pruthwik Mishra IIIT, Hyderabad
Sayantan Mitra, IIT Patna, India
Dr. Vinay Kumar Mittal, Professor, KL University
Ashutosh Modi, Indian Institute of Technology Kanpur
Aditya Mogadala, Saarland University
Vandan Mujadia, IIIT-Hyderabad
Animesh Mukherjee, IIT Kharagpur
Aditi Mukherjee, IIIT-Hyderabad
Siddhartha Mukherjee, Samsung R&D Institute India, Bangalore
Abhijith Athreya Mysore Gopinath, Pennsylvania State University
Abhishek Narayanan, Department of Computer Science and Engineering, PES University
Mukuntha Narayanan SundararamanNarayanan Sundararaman, IIT Patna, India
Sudip Kumar Naskar, Jadavpur University
Tapas Nayak, National University of Singapore
Hamada Nayel, Benha University
Anish Nediyanchath, Samsung R&D Institute India - Bangalore
Vasudevan Nedumpozhimana, TU Dublin
Preksha Nema, IIT Madras
Kishorjit Nongmeikapam, Indian Institute of Information Technology(IIIT) Manipur
Deepak P, Queen's University Belfast
Jisha P Jayan, IIITMK
Partha Pakray, National Institute of Technology Silchar
Santanu Pal, Saarland University
Girish Palshikar, Tata Consultancy Services Limited
Rrubaa Panchendrarajan, National University of Singapore
Shantipriya Parida, Idiap Research Institute
Md. Aslam Parwez, Jamia Millia Islamia
Tanvina Patel, Cogknit Semantics
Kevin Patel, IIT Bombay
Sangameshwar Patil, TRDDC, TCS Research, and Innovation
Braja Gopal Patra, Department of Population Health Sciences, Weill Cornell Medicine
Abhipsa Patro, IIIT Bhubaneswar
Sayanta Paul, Indian Institute of Management
Jyoti Pawar, Goa University, Goa
Sachin Pawar, Tata Consultancy Services Ltd.
Jerin Philip, Naver Labs Europe
Soujanya Poria, Singapore University of Technology and Design
Suhan Prabhu, International Institute of Information Technology, Hyderabad
Ganesh Prasad, Samsung
S R Mahadeva Prasanna, IIT Dharwad
Michal Ptaszynski, Kitami Institute of Technology
Srinivas PYKL, IIIT Sri City
Vartika Rai, IIIT Hyderabad

Sai Krishna Rallabandi, Carnegie Mellon University
Surangika Ranathunga, university of moratuwa
Hanumant Redkar, IIT Bombay
Pattabhi RK Rao, AU-KBC Research center
Paolo Rosso, Universitat Politecnica de Valencia
Vijay Rowtula, International Institute of Information Technology, Hyderabad
Pradeep Kumar Roy, Vellore Institute of Technology, Vellore
Aniruddha Roy, IIT Kharagpur
Sowmya S Sundaram, IIT Madras
Tulika Saha, IIT Patna
Atanu Saha, Jadavpur university
Sujan Kumar Saha, Dept. of CSE, Birla Institute of Technology Mesra
Saumajit Saha, TCS Research and Innovation Labs
Sovan Kumar Sahoo, IIT Patna
Pracheta Sahoo, The University of Texas at Dallas
Tanik Saikh, IIT Patna
Naveen Saini, IIT Patna
Suyash Sangwan, IIT Patna
Karthik Sankaranarayanan, IBM Reseach
Sebastin Santy, Microsoft Research
Kamal Sarkar, Computer Science and Engineering Department, Jadavpur University
Sandip Sarkar, Hijli College
Sunil Saumya, IIIT Dharwad
Moritz Schaeffer, Johannes Gutenberg University of Mainz
Peter Scharf, International Institute of Information Technology; Indian Institute of Advanced Study
Sanket Shah, MSRI
Raksha Sharma, IIT Roorkee
Ravi Shekhar, Queen Mary University of London
Manish Shrivastava, International Institute of Information Technology Hyderabad
Smriti Singh, Samsung Research UK
Thoudam Doren Singh, NIT Silchar
Sandhya Singh, IITB
Vikram Singh, IIT Patna
Jyoti Prakash Singh, NIT Patna
Pardeep Singh, Jawaharlal Nehru University
Manjari Sinha, IITKharagpur
Sunayana Sitaram, Microsoft Research India
Saurabh Srivastava,TCS Research
Keh-Yih Su, Institute of Information Science, Academia Sinica
Chanchal Suman, IIT Patna
Bapi Raju Surampudi, International Institute of Information Technology Hyderabad
Partha Talukdar, Indian Institute of Science
Gaurish Thakkar, University of Zagreb
Anil Thakur, IIT (BHU) Varanasi
Nidhi Thakur, IIT Patna
Medari Tham, Assam Don Bosco University
Uthayasanker Thayasivam, University of Moratuwa
Uma Shanker Tiwary, IIIT Allahabad
Prajna Upadhyay, IIT Delhi

Ashwini Vaidya, IIT Delhi
Shalaka Vaidya, IIIT Hyderabad
Vasudeva Varma, IIIT Hyderabad
Deeksha Varshney, IIT Patna
Ashraf Kamal, Jamia Millia Islamia
Sriram Venkatapathy, Amazon
Samudra Vijaya, IIT Guwahati
Anil Kumar Vuppala, IIIT Hyderabad
Saumitra Yadav, International Institute of Information Technology, Hyderabad

Organized by:

Indian Institute of Technology Patna

Natural Language Processing Association of India

Table of Contents

Conference Program

Day 1: Saturday, December 19, 2020

+ 10:00 - 11:00 **Inaugural Ceremony**

+ 11:30 - 13:00 BREAK

+ 13:00 -14:30 **Technical Session I:** Information Extraction-I
Session Chair: Karthik Sankaranarayanan

Automatic Hadith Segmentation using PPM Compression
Taghreed Tarmom, Eric Atwell and Mohammad Alsalka

Learning to Interact: An Adaptive Interaction Framework for Knowledge Graph Embeddings
Chandrahas, Nilesh Agrawal and Partha Talukdar

Event Argument Extraction using Causal Knowledge Structures
Debanjana Kar, Sudeshna Sarkar and Pawan Goyal

Weak Supervision using Linguistic Knowledge for Information Extraction
Sachin Pawar, Girish Palshikar, Ankita Jain, Jyoti Bhat and Simi Johnson

Technical Session II: NLP Language Documentation and Preservation
Session Chair: Sobha Lalitha Devi

A Grammatical Sketch of Asur: A North Munda language
Zoya Khalid

Treatment of optional forms in Mathematical modelling of Pāṇini
Anupriya Aggarwal and Malhar Kulkarni

Language Model Metrics and Procrustes Analysis for Improved Vector Transformation of NLP Embeddings
Thomas Conley and Jugal Kalita

Assamese Word Sense Disambiguation using Genetic Algorithm
Arjun Gogoi, Nomi Baruah and Shikhar Kr. Sarma

Technical Session III: Computational Social Science and Social Media
Session Chair: Tanmoy Chakraborty

Identifying Complaints from Product Reviews in Low-resource Scenarios via Neural Machine Translation
Raghvendra Pratap Singh, Rejwanul Haque, Mohammed Hasanuzzaman and Andy Way

Hater-O-Genius Aggression Classification using Capsule Networks
Parth Patwa, Srinivas PYKL, Amitava Das, Prerana Mukherjee and Viswanath Pulabaigari

Native-Language Identification with Attention
Stian Steinbakken and Björn Gambäck

Acoustic Analysis of Native (L1) Bengali Speakers' Phonological Realization of English Lexical Stress Contrast
Shambhu Nath Saha and Shyamal Kr. Das Mandal

+15:00-16:00 **Keynote Lecture 1:** Prof. Dr. Iryna Gurevych, Technische Universität Darmstadt, Germany
Title: Let's Argue - Understanding and Generating Natural Language Arguments
Session Chair: Sudeshna Sarkar

+16:00-16:30 **BUFFER**

Technical Session IV: Sentiment and Emotion Analysis
Session Chair: Amitava Das

Polarization and its Life on Social Media: A Case Study on Sabarimala and Demonetisation
Ashutosh Ranjan, Dipti Sharma, and Radhika Krishnan

Only text? only image? or both? Predicting sentiment of internet memes
Pranati Behera, Mamta and Asif Ekbal

Leveraging Multi-domain, Heterogeneous Data using Deep Multitask Learning for Hate Speech Detection
Prashant Kapil and Asif Ekbal

CLPLM: Character Level Pretrained Language Model for Extracting Support Phrases for Sentiment Labels
Raj Pranesh, Sumit Kumar and Ambesh Shekhar

Technical Session V: Named Entity Recognition
Session Chair: Sriparna Saha

The WEAVE Corpus: Annotating Synthetic Chemical Procedures in Patents with Chemical Named Entities
Ravindra Nittala and Manish Shrivastava

Deep Neural Model for Manipuri Multiword Named Entity Recognition with Unsupervised Cluster Feature
Jimmy Laishram, Kishorjit Nongmeikapam and Sudip Kumar Naskar

Constructing a Korean Named Entity Recognition Dataset for the Financial Domain using Active Learning
Dong-Ho Jeong, Min-Kang Heo, Hyung-Chul Kim and Sang-Won Park

Technical Session VI: Multimodality/Speech Recognition
Session Chair: Anil Kumar Vuppala

A Multi-modal Personality Prediction System
Chanchal Suman, Aditya Gupta, Sriparna Saha and Pushpak Bhattacharyya

End-to-End Automatic Speech Recognition for Gujarati
Deepang Raval, Vyom Pathak, Muktan Patel and Brijesh Bhatt

Using multiple ASR hypotheses to boost i18n NLU performance
Charith Peris, Gokmen Oz, Khadige Abboud, Venkata sai Varada, Prashan Wanigasekara, and Haidar Khan

Leveraging Latent Representations of Speech for Indian Language Identification
Samarjit Karmakar and P Radha Krishna

+17:30-18:00 **BUFFER**

+18:00-19:00 **Keynote Lecture 2:** Prof. Eduard Hovy, Carnegie Mellon University
Title: From Simple to Complex QA
Session Chair: Prof. Pushpak Bhattacharyya

+19:00-19:30 **BUFFER**

+19:30-21:00 **NLPAI Meeting**

Day 2: Sunday, December 20, 2020

+ 11:00 -13:00 **Technical Session VII:** Information Extraction-II
Session Chair: Dipankar Das

Inducing Interpretability in Knowledge Graph Embeddings
Chandrahas, Tathagata Sengupta, Cibi Pragadeesh and Partha Talukdar

Solving Arithmetic Word Problems Using Transformer and Pre-processing of Problem Texts
Kaden Griffith and Jugal Kalita

Generative Adversarial Networks for Annotated Data Augmentation in Data Sparse NLU
Olga Golovneva and Charith Peris

Semantic Extractor-Paraphraser based Abstractive Summarization
Anubhav Jangra, Raghav Jain, Vaibhav Mavi, Sriparna Saha and Pushpak Bhattacharyya

Technical Session VIII:: Machine Learning Applications to NLP-I
Session Chair: Aditya Joshi

BertAA : BERT fine-tuning for Authorship Attribution
Maël Fabien, Esaú Villatoro-Tello, Petr Motlicek and Shantipriya Parida

Claim extraction from text using transfer learning.
Acharya Ashish Prabhakar, Salar Mohtaj and Sebastian Möller

On-Device detection of sentence completion for voice assistants with low-memory footprint
Rahul Kumar, Vijeta Gour, Chandan Pandey, Godawari Sudhakar Rao, Priyadarshini Pai, Anmol Bhasin and Ranjan Samal

A New Approach to Claim Check-Worthiness Prediction and Claim Verification
Shukrity Si, ANISHA DATTA and Sudip Kumar Naskar

Technical Session IX: Machine Learning Applications to NLP-II
Session Chair: Ashutosh Modi

Clickbait in Hindi News Media : A Preliminary Study
Vivek Kaushal and Kavita Vemuri

Does a Hybrid Neural Network-based Feature Selection Model Improve Text Classification?
Suman Dowlagar and Radhika Mamidi

Semantic Slot Prediction on low corpus data using finite user-defined list
Bharatram Natarajan, Dharani Simma, Chirag Singh, Anish Nediyanchath and Sreoshi Sengupta

Detection of Similar Languages and Dialects Using Deep Supervised Autoencoder
Shantipriya Parida, Esaú Villatoro-Tello, Sajit Kumar, Maël Fabien and Petr Motlicek

Sentimental Poetry Generation
Kasper Aalberg Røstvold and Björn Gambäck

Towards Performance Improvement in Indian Sign Language Recognition
Kinjal Mistree, Devendra Thakor and Brijesh Bhatt

Technical Session X: Machine Translation-I
Session Chair: Nikesh Garera

Exploring Pair-Wise NMT for Indian Languages
Kartheek Akella, Sai Himal Allu, Sridhar Suresh Ragupathi, Aman Singhal, Zeeshan Khan, C.V. Jawahar and Vinay P. Namboodiri

PhraseOut: A Code Mixed Data Augmentation Method for Multilingual Neural Machine Tranlsation
Binu Jasim, Vinay Namboodiri and C V Jawahar

Efforts Towards Developing a Tamang Nepali Machine Translation System
Binaya Kumar Chaudhary, Bal Krishna Bal and Rasil Baidar

TREE ADJOINING GRAMMAR BASED "LANGUAGE INDEPENDENT GENERATOR"
Pavan Kurariya, Prashant Chaudhary, Jahnavi Bodhankar, Lenali Singh, Ajai Kumar and Hemant Darbari

+13:00-14:00 **BREAK**

+ 14:00 -16:00 **Technical Session XI:** Language Resources- I
Session Chair: Girish Palsikar

A Rule Based Lightweight Bengali Stemmer
Souvick Das, Rajat Pandit and Sudip Kumar Naskar

Towards Bengali Word Embedding: Corpus Creation, Intrinsic and Extrinsic Evaluations
Md. Rajib Hossain and Mohammed Moshiul Hoque

Annotated Corpus of Tweets in English from Various Domains for Emotion Detection
Soumitra Ghosh, Asif Ekbal, Pushpak Bhattacharyya, Sriparna Saha, Vipin Tyagi, Alka Kumar, Shikha Srivastava and Nitish Kumar

STHAL: Location-mention Identification in Tweets of Indian-context
Kartik Verma, Shobhit Sinha, Md. Shad Akhtar and Vikram Goyal

Developing a Faroese PoS-tagging solution using Icelandic methods
Hinrik Hafsteinsson and Anton Karl Ingason

Technical Session XII: Language Resources-II
Session Chair: Manish Srivastava and Vishal Goyal

Increasing accuracy of a semantic word labelling tool based on a small lexicon
Hugo Sanjurjo-González

Creation of Corpus and Analysis in Code-Mixed Kannada-English Social Media Data for POS Tagging
Abhinav Reddy Appidi, Vamshi Krishna Srirangam, Darsi Suhas and Manish Shrivastava

Exploration of Cross-lingual Summarization for Kannada-EnglishLanguage Pair
Vinayaka R Kamath, Rachana Aithal K R, Vennela K and Mamatha HR

SUKHAN: Corpus of Hindi Shayaris annotated with Sentiment Polarity Information
Salil Aggarwal

Technical Session XIII: NLP for Education
Session Chair: Sudip Naskar

Cognitively Aided Zero-Shot Automatic Essay Grading
Sandeep Mathias, Rudra Murthy, Diptesh Kanojia and Pushpak Bhattacharyya

Automated Arabic Essay Evaluation
Abeer Alqahtani and Amal Alsaif

Question and Answer pair generation for Telugu short stories
Meghana Bommadi, Shreya Terupally and Radhika Mamidi

ScAA: A Dataset for Automated Short Answer Grading of Children's free-text Answers in Hindi and Marathi
Dolly Agarwal, Somya Gupta and Nishant Baghel

+16:00-16:30 **BUFFER**

+16:30-17:30 **Technical Session XIV:** Information Retrieval and Text Mining
Session Chair: Sudeshna Sarkar

Improving Passage Re-Ranking with Word N-Gram Aware Coattention Encoder
Chaitanya Alaparthi and Manish Shrivastava

Self-Supervised Claim Identification for Automated Fact Checking
Archita Pathak, Mohammad Abuzar Shaikh and Rohini Srihari

D-Coref: A Fast and Lightweight Coreference Resolution Model using DistilBERT
Chanchal Suman, Jeetu Kumar, Sriparna Saha and Pushpak Bhattacharyya

Technical Session XV: Syntax
Session Chair: Pawan Goyal

Self Attended Stack-Pointer Networks for Learning Long Term Dependencies
Salih Tuc and Burcu Can

ThamizhiUDp: A Dependency Parser for Tamil
Kengatharaiyer Sarveswaran and Gihan Dias

Free Word Order in Sanskrit and Well-nestedness
Sanal Vikram and Amba Kulkarni

Technical Session XVI: Machine Translation-II
Session Chair: Anoop Kunchukuttan

English to Manipuri and Mizo Post-Editing Effort and its Impact on Low Resource Machine Translation
Loitongbam Sanayai Meetei, Thoudam Doren Singh, Sivaji Bandyopadhyay, Mihaela Vela and Josef van Genabith

Improving Neural Machine Translation for Sanskrit-English
Ravneet Punia, Aditya Sharma, Sarthak Pruthi and Minni Jain

Leveraging Alignment and Phonology for low-resource Indic to English Neural Machine Transliteration
Parth Patel and Manthan Mehta

+17:30-18:00 **BUFFER**

+18:00-19:00 **Keynote Lecture 3:** David Yarowsky, John Hopkins University, USA
Title: Translingual Learning of 1000+ Languages
Session Chair: Prof. Dipti Misra Sharma

+19:00-19:30 **Valedictory Session**

Pre-conference: Friday, December 18, 2020

+ 15:00 -16:30 **Doctoral Consortium Session**
Session Chair: Anil Kumar Singh

Parsing Indian English News Headlines
Samapika Roy, Sukhada and Anil Kumar Singh

WORD SENSE DISAMBIGUATION FOR KASHMIRI LANGUAGE USING SUPERVISED MACHINE LEARNING
TAWSEEF AHMAD MIR

WEKA in Forensic Authorship Analysis: A corpus-based approach of Saudi Authors
Mashael AlAmr and Eric Atwell

The WEAVE Corpus: Annotating Synthetic Chemical Procedures in Patents with Chemical Named Entities

Ravindra Nittala
Language Technology Research Centre,
IIIT - Hyderabad, India
ravindra.n@research.iiit.ac.in

Manish Shrivastava
Language Technology Research Centre,
IIIT - Hyderabad, India
m.shrivastava@iiit.ac.in

Abstract

The modern pharmaceutical industry depends on the iterative design of novel synthetic routes for drugs while not infringing on existing intellectual property rights. Such a design process calls for analyzing many existing synthetic chemical reactions and planning the synthesis of novel chemicals. These procedures have been historically available in unstructured raw text form in publications and patents. To facilitate automated synthetic chemical reactions analysis and design the novel synthetic reactions using Natural Language Processing (NLP) methods, we introduce a Named Entity Recognition (NER) dataset of the Examples section in 180 full-text patent documents with 5188 synthetic procedures annotated by domain experts. All the chemical entities which are part of the synthetic discourse were annotated with suitable class labels. We present the second-largest chemical NER corpus with 100,129 annotations and the highest IAA value of 98.73% (F-measure) on a 45 document subset. We discuss this new resource in detail and highlight some specific challenges in annotating synthetic chemical procedures with chemical named entities. We make the corpus available to the community to promote further research and development of downstream NLP systems applications. We also provide baseline results for the NER model to the community to improve on.

1 Introduction

There is a renewed interest in academia and industry to access the information regarding chemical and chemical reactions currently available in unstructured raw text form in journal publications and patents (Coley et al., 2017; Segler et al., 2018; Mysore et al., 2019) using machine learning. Also, several chemical NER datasets exist. With increasing demand in automated chemical synthesis design and planning novel chemical reactions, we need to shift away from the annotation of title and abstract of patents or reactions in isolation to the patents' core, the Examples section. The CHEMDNER-patents corpus (Krallinger et al., 2015c) is the only dataset focusing on titles and abstracts. The Chapati corpus (Grego et al., 2009) and BioSemnatics corpus (Akhondi et al., 2014) focus on the full text of patents for annotation. The reason for the insufficiency of these corpora is discussed in detail in Section 3.3 and 3.5. The ChEMU labs introduced a named entity dataset with chemical role labels (Nguyen et al., 2020). As part of the dataset, they have annotated only snippets of reaction text from the patents' experimental section. They also acknowledge the problem of entity often referring to context beyond the current reaction text[1]. This context cannot be accounted for by the snippets of reaction text in isolation. As part of the WEAVE corpus, we would like to annotate the chemical entities in their full reaction discourse. This would enable us to model the context beyond the immediate reaction text. We refer readers to the supporting information containing full-text patents to understand how the discourse varies from section to section.

A patent is the grant of a legal right by a patent office to an inventor. This grant provides the inventor exclusive rights for a designated period of time in exchange for comprehensive invention disclosure. The disclosure should be complete, such that a person well versed in the field should be able to reproduce this patented process, design, or invention. This disclosure is done in the Examples section of a patent. Hence the Examples section is fundamentally different in its linguistic structure from other sections in a patent. It is the most useful part of understanding the synthetic chemical

[1] https://chemu-patent-ie.github.io/resources/Annotation_Guidelines_CLEF2020_ChEMU_task1.pdf

Proceedings of the 17th International Conference on Natural Language Processing, pages 1–9
Patna, India, December 18 - 21, 2020.

reactions given in the patent.

1.1 Related work

There is a large body of chemical and biomedical NER literature. We refer readers to Yadav and Bethard (2018) and Huang et al. (2020) for a comprehensive survey. We include a summary of the publicly available datasets as follows: Chapati corpus (Grego et al., 2009) is a manually annotated set of 40 patents with 11,162 annotations. The chemical named entities identified were mapped to the Chemical Entities of Biological Interest (ChEBI) database. BioSemantics corpus (Akhondi et al., 2014) is a manually annotated set of patents. This corpus has two sets: First, a harmonized set of 47 patents with 36,537 annotations, and the second set of 198 patents with 400,125 annotations. Besides chemical entity mentions, they also annotate diseases, targets, modes of actions (MOAs), OCR errors, and spelling errors. It is the largest chemical NER dataset. BC-IV CHEMDNER corpus (Krallinger et al., 2015a) is an annotated set of 10,500 titles and abstracts from the PubMed database with 84,355 annotations. BC-V CHEMDNER-patents corpus (Krallinger et al., 2015c) is an annotated set of 21,000 titles and abstracts from patents with 99,634 annotations. With BC-IV CHEMDNER corpus and BC-V CHEMDNER-patents corpus being the widely cited among these. CHEMDNER-patents corpus exclusively focuses on chemical entity mentions. The entity mention classes are a variant of earlier published CHEMDNER corpus (Krallinger et al., 2015b). Nguyen et al. (2020) have introduced a new evaluation lab named ChEMU. It focuses on two tasks: First, named entity recognition of chemical compounds and assign the compound's role within a chemical reaction. Second, event trigger detection and argument identification of previously detected chemical entities. In the publically available NER dataset[2], there are 20,186 annotations (train + dev) in 1125 reaction snippets extracted from 170 patents.

1.2 Structure of a patent

A typical US patent[3] granted has the following discourse structure: Patent grant number, Title, Bibliography, Abstract, Other Patent Relations, Brief Summary, Detailed Description, and Claims. The intellectual property rights or the innovative part of the patent granted resides in the examples contained in the Detailed Description section. This section will be analyzed thoroughly for any novel synthetic route to be non-infringing on existing intellectual property rights. Therefore in the next section, we present the WEAVE[4] patents corpus, which focuses exclusively on synthetic procedures in the Examples section.

2 The WEAVE patents corpus

An important consideration in preparing a corpus for NER training, development, and evaluation sets is selecting documents representing the distribution of chemical named entities seen in related documents. In the WEAVE corpus, the focus is on synthetic chemical procedures and the chemical entities present. Two considerations influenced document selection in our corpus. First, the documents used in the corpus should be available without copyright protection. Second, they are complementary to existing datasets. We accessed the patents from the United States Patent and Trademark Office (USPTO)[5]. Following criterion were applied to further subset the patents for annotation:

- **IPC code**: The selection of patents for the WEAVE corpus was made based on IPC (International Patent Classification) code. Patents which belonged to at least A61K (Preparations for Medical, Dental, or Toilet purposes)[6] or C07D (Heterocyclic compounds) [7] were selected. This enriched patents with chemical entities in medicinal and organic chemistry. An additional criterion for selection within this subset was the presence of synthetic organic procedures.

- **Date and Publication type**: We decided to select patents that were granted in the years

[2] http://chemu.eng.unimelb.edu.au/

[3] USPTO, https://www.uspto.gov

[4] to form something from several different things or to combine several different things, in a complicated or skilled way https://dictionary.cambridge.org/dictionary/english/weave

[5] USPTO Bulk Data Storage System (BDSS) https://bulkdata.uspto.gov/#pats

[6] https://www.wipo.int/classifications/ipc/en/ITsupport/Version0170101/transformations/ipc/20170101/en/htm/A61K.htm

[7] https://www.wipo.int/classifications/ipc/en/ITsupport/Version0170101/transformations/ipc/20170101/en/htm/C07D.htm

2018 and 2019. This would ensure the availability of patents in XML format and text free from OCR errors.

- **Character encoding and language**: XML character entities were converted to corresponding UTF-8 characters, and the full text was encoded in UTF-8 encoding. As the patents were selected from USPTO, only English language patents were included.

- **Document format**: The patent in XML format was converted to a UTF-8 encoded text file. Only the paragraph elements, headings, subheadings, and tables were written to the text file. All the formatting elements like bold, italics, subscript, and superscript were discarded. Bibliographic details and XML formatting was also discarded. There was no restriction on the number of lines in documents.

- **Documents inclusion and exclusion**: Patents covering Inorganic, Organometallic, Polymers, Natural products, Proteins, DNA/RNA, Polymorphic crystal forms were excluded. The overriding criterion for inclusion was at least one synthetic organic procedure in the Examples section, and this was manually checked in each document.

- **Final document sets**: After applying the above selection criteria and prepossessing, we were left with 180 documents. The summary of these sets is given in Table 1. These were randomly assigned to training, development, and test sets. 45 documents from the above settings were used for the Inter-annotator agreement (IAA). For display performance in BRAT, all patents were split into files of 100 lines each before annotation and later concatenated into a single document after annotation.

Set	Documents	Reactions
Evaluation	45	438
Training	60	1311
Development	60	2020
Test	60	1857
Overall	180	5188

Table 1: Document sets. Evaluation set is a subset of overall 180 documents

3 Corpus annotation

3.1 Annotation tools

Neves and Leser (2012) have surveyed the annotation tools available for biomedical literature. They determined that BRAT was easy to use and customizable as per the annotation scheme among the tools reviewed. Hence we used the BRAT Rapid Annotation Tool (BRAT) (Stenetorp et al., 2012) for the entire annotation process and BRAT standoff format for storing the annotations.

3.2 Evaluation metric

We used CoNLL 2003 shared task (Tjong Kim Sang and De Meulder, 2003) evaluation script to compute the macro averaged F-measure on named entity annotations. The annotation output in the BRAT standoff format was converted to CoNLL 2003 shared task format with BIO tagging representation before computing the F-measure. We used F-measure as the evaluation metric for IAA as suggested by Corbett et al. (2007) and Kolarik et al. (2008). CoNLL 2003 shared task evaluation script evaluates an entity to be valid by matching the chemical mention and class label. The use of F-measure provides an advantage in a direct comparison between system performance and inter-annotator agreement (Grouin and Névéol, 2014).

3.3 Annotation scheme

We had to make a choice of designing our own scheme or utilize an existing scheme. Based on publicly available guidelines and corpora, we had a choice between Chapati corpus by Chemical Entities of Biological Interest (ChEBI) and European Patent Office (EPO) (Grego et al., 2009), BioSematics corpus (Akhondi et al., 2014), CHEMDNER corpus (Krallinger et al., 2015a), CHEMDNER-patents corpus (Krallinger et al., 2015c) and ChEMU Labs NER corpus (Nguyen et al., 2020). In Chapati corpus, 40 patents were manually annotated with 11,162 annotations (Grego et al., 2009). The number of annotated patents and the corresponding number of annotations was small in size.

We were left with a choice between BioSemantics, CHEMDNER, and CHEMDNER-patents corpora. On a closer look at BioSemantics corpus, which was based on 15 rules published in their article (Akhondi et al., 2014), we noticed that the IAA (F-score), when considered for only chemical mentions in the corpus, varies from 0.94 to 0.38 depending on entity type and the agreement between the

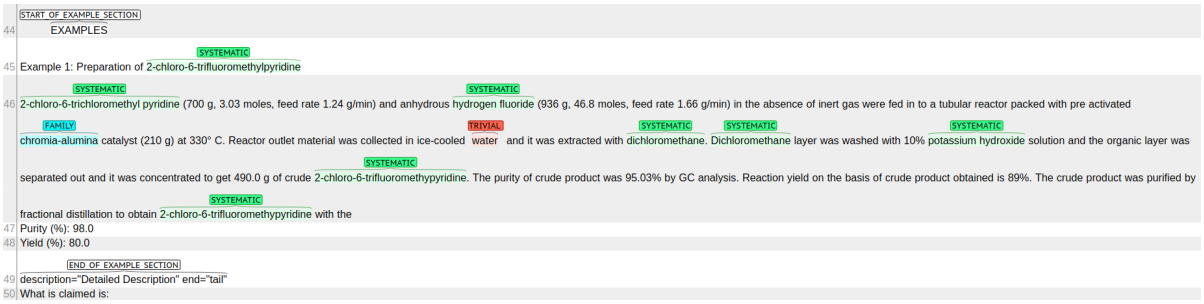


Figure 1: An example an annotated organic reaction, within the Examples section of patent.

four annotator groups on the harmonized patents set (47 patents) (Akhondi et al., 2014). The wide variation in IAA indicates a lack of consistency in guidelines and the need for multiple disambiguation steps. This could be potentially misleading to the annotators.

The near-simultaneous publication of the ChEMU Labs NER dataset[8] (Nguyen et al., 2020) with this publication precluded a full evaluation of the dataset. After reviewing the guidelines[9], it was determined that this dataset is not suitable for the chemical named entity recognition in the full discourse of reaction text in the Examples section.

The extensive guidelines documentation (30 pages), illustrated with examples, led us to choose the annotation scheme developed for BioCreative IV (BC-IV) CHEMDNER (Krallinger et al., 2015a). As modified in BioCreative V (BC-V), CHEMDNER-patents task (Krallinger et al., 2015c) to be used for the WEAVE corpus annotation process. CHEMDNER-patents task had annotated titles and abstracts from 21,000 patents with 99,625 annotations (Krallinger et al., 2015c). SYSTEMATIC, IDENTIFIER, FORMULA, TRIVIAL, ABBREVIATION (ABBV), FAMILY and MULTIPLE entity mention classes as reported by Krallinger et al. (2015c) were utilized. We chose to annotate the Examples section of the patent with synthetic organic procedures against title and abstract only in the CHEMDNER-patents task (Krallinger et al., 2015c). This is illustrated in Figure 1.

3.4 Annotation process

The entire annotation process was done in two stages. The first stage work was done to establish the inter-annotator agreement on the evaluation set of 45 documents. The documents were annotated by nine chemistry domain experts with no formal linguistics experience and were equally divided between them (5 each).

These 45 documents were independently double annotated by another chemistry domain expert, designated as lead annotator with formal linguistics experience. The lead annotator's annotations were designated as the gold standard for evaluating the quality of annotation by the nine annotators. These 45 documents were compared to the gold standard using F-measure. Once the annotation consistency was established, the second stage work was done on the rest of the 135 documents. With each annotator getting 15 documents. Following the concept of annotator-reviser (or adjudicator) agreement (Campillos et al., 2018; Bada et al., 2012), annotators were free to consult the lead annotator throughout the annotation process regarding guidelines.

3.5 IAA statistics

CLASS	Precision	Recall	F1
ABBV.	98.50%	99.88%	99.19
FAMILY	90.86%	97.28%	93.96
FORMULA	98.84%	95.63%	97.21
IDENTIFIER	80.00%	72.73%	76.19
MULTIPLE	75.00%	100.00%	85.71
SYSTEMATIC	99.02%	99.13%	99.07
TRIVIAL	98.85%	100.00%	99.42
Overall	98.66%	98.81%	98.73

Table 2: IAA statistics.

Table 2 presents IAA statistics for 45 documents set. The average F-measure was 98.73%. Bada et al. (2012) have reported 90+% IAA level following the annotator-reviser (or adjudicator) agreement concept. Hence the F-measure reported by us is

[8] http://chemu.eng.unimelb.edu.au/

[9] https://github.com/chemu-patent-ie/chemu-patent-ie.github.io/tree/master/resources/Annotation_Guidelines_CLEF2020_ChEMU_task1.pdf

consistent with published results. This IAA value is the highest reported to date on the chemical entity mention dataset. The F-measure at the micro-level was the lowest for IDENTIFIER (76.19%) and MULTIPLE (85.71%). This can be attributed to the data sparsity in the corpus for these two classes. Tables 4, 5 and 6 demonstrate that the data sparsity for these two classes can also be seen in BC-IV CHEMDNER (Krallinger et al., 2015a) and BC-V CHEMDNER-patents task (Krallinger et al., 2015c).

Akhondi et al. (2014) have reported an annotated chemical patent corpus, which besides chemical mentions, also annotates diseases, protein targets, and MOAs in the patents. The best-reported IAA value among a set of values was 78% (F-score). Krallinger et al. (2015b) in BC-IV CHEMDNER task has reported the IAA value of 91% (F-score) while matching the chemical mention ignoring the class label. When the class label was also considered, the IAA value was 85.26% (F-score). Krallinger et al. (2015c) in BC-V CHEMDNER-patents task have not reported any IAA value and have proposed an IAA study based on a blind annotation of 200 patent abstracts in case of the chemical entity mentions. To the best of our knowledge, this has not yet been published.

Despite no published IAA study for CHEMDNER-patents corpus, we relied on the extensive guidelines published as part of their corpus.

3.6 Error Analysis

Table 3 presents the error analysis of the doubly annotated 45 documents. In the table, rows represent the gold standard labels, and columns represent the annotator's labels. Of the 7503 gold labels, 90 labels (1.2%) were assigned outside the reaction discourse. These should have been assigned to the OTHER class. 78 labels (1.0%) where they should have been assigned one of seven class labels, they were assigned, OTHER class. Only 4 (0.05%) were assigned the incorrect label within the seven class labels.

The error analysis demonstrates that annotators were able to assign the class labels to the chemical entities. The majority of the errors occurred at the boundary of reaction discourse. These errors were communicated to the annotators. They were trained to identify the reaction discourse boundaries and the chemical entities present. They were also encouraged to consult the lead annotator in case of any doubt.

3.7 Corpus statistics

Tables 4, 5 and 6 present the counts of chemical entity mention class labels in the WEAVE corpus (180 documents). These were randomly divided into Training, Development, and Test sets and compared with similar counts from BC-IV CHEMDNER (Krallinger et al., 2015a) and BC-V CHEMDNER-patents task (Krallinger et al., 2015c). Table 7 presents the statistics for the counts of annotations in the WEAVE corpus and CHEMDNER-patents corpus. There are a total of 100,129 annotations with an average of 556 annotations per document. As shown in the table, there is a wide variation between average and median counts per document. This skew is due to a small number of documents having a large number of annotations (Bada et al., 2012). This assertion is supported by the minimum and maximum count across 180 documents.

The top three entity mention classes as a percentage of total annotations in WEAVE corpus was: SYSTEMATIC (49.73%), FORMULA (26.58%), and ABBREVIATION (11.25%). The corresponding distribution of the top three classes in BC-IV CHEMDNER task was: SYSTEMATIC (30.36%), TRIVIAL (22.69%) and ABBREVIATION (15.55%), and in BC-V CHEMDNER-patents task was: FAMILY (36.49%), SYSTEMATIC (28.79%) and TRIVIAL (26.11%). The statistical distribution of entities mentions classes between WEAVE corpus and CHEMDNER-patents corpus is different. Hence the need for annotation of the Examples section of patents was felt. This would significantly help develop machine learning models tailored for the Examples section and downstream processing of synthetic organic reactions in patents.

4 Experiments

To establish some baseline performance parameters for the evaluation of the WEAVE corpus, we applied the NER model[10] developed by Yadav et al. (2018), which has been successfully applied in Multilingual, Clinical and Drug NER. Morphological features have been successfully applied in named entity recognition. In submissions to BC-IV CHEMDNER task (Krallinger et al., 2015a)

[10] https://github.com/vikas95/Pref_Suff_Span_NN

	ABBV.	FAMILY	FORMULA	IDENTIFIER	MULTIPLE	SYSTEMATIC	TRIVIAL	OTHER
ABBV.	855	1	0	0	0	0	0	0
FAMILY	0	179	0	0	0	0	0	5
FORMULA	2	0	854	0	0	0	0	37
IDENTIFIER	0	0	0	8	0	0	1	2
MULTIPLE	0	0	0	0	6	0	0	0
SYSTEMATIC	0	0	0	0	0	4658	0	34
TRIVIAL	0	0	0	0	0	0	861	0
OTHER	11	17	10	2	2	39	9	498807

Table 3: Error analysis of annotations.

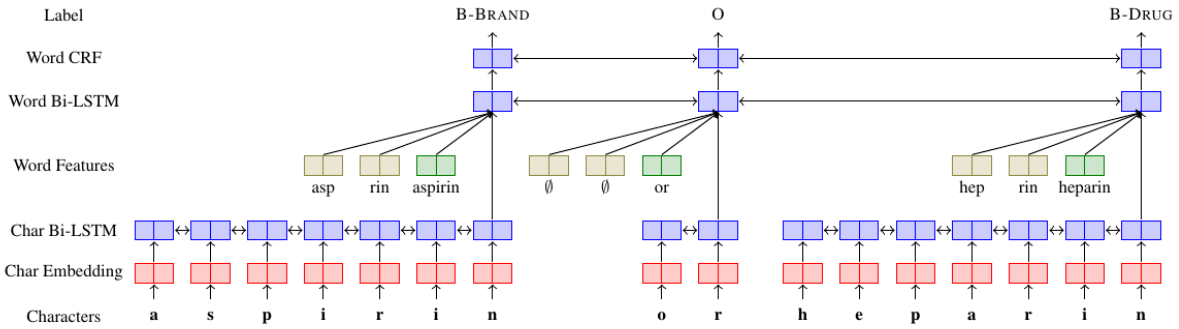


Figure 2: Architecture of NER model proposed by Yadav et al. (2018)

CLASS	BC-IV	BC-V	WEAVE
ABBV.	4538	588	2520
FAMILY	4090	12209	783
FORMULA	4448	2239	6709
IDENTIFIER	672	99	47
MULTIPLE	202	140	6
NO CLASS	40	-	-
SYSTEMATIC	6656	9570	14547
TRIVIAL	8832	8698	2756
Total	29478	33543	27368

Table 4: Training set.

CLASS	BC-IV	BC-V	WEAVE
ABBV.	4521	454	3857
FAMILY	4223	11710	769
FORMULA	4137	2120	9679
IDENTIFIER	639	125	47
MULTIPLE	188	141	13
NO CLASS	32	-	-
SYSTEMATIC	6816	9194	20106
TRIVIAL	8970	8398	4054
Total	29526	32142	38525

Table 5: Development set.

and BC-V CHEMDNER-patents task (Krallinger et al., 2015c) they feature prominently in the top-performing models.

4.1 Word embeddings

200-dimension GloVe embeddings (Pennington et al., 2014) were trained on text extracted from 100,000 US patents belonging to IPC code A61K[11] and C07D[12]. A window of word co-occurrence of

[11] https://www.wipo.int/classifications/ipc/en/ITsupport/Version0170101/transformations/ipc/20170101/en/htm/A61K.htm

[12] https://www.wipo.int/classifications/ipc/en/ITsupport/Version0170101/transformations/ipc/20170101/en/htm/C07D.htm

CLASS	BC-IV	BC-V	WEAVE
ABBV.	4059	331	4892
FAMILY	3622	12319	597
FORMULA	3443	2459	10231
IDENTIFIER	513	54	57
MULTIPLE	199	137	10
NO CLASS	41	-	-
SYSTEMATIC	5666	9818	15145
TRIVIAL	7808	8831	3304
Total	25351	33949	34236

Table 6: Test set.

Type	WEAVE	BC-V
Total annotations	100,129	99,634
Average per document	522	5
Median per document	366	3
Minimum per document	10	0
Maximum per document	3640	233

Table 7: Statistics for counts of annotations

8 and word frequency of 1 was used to train the uncased text. The resulting embeddings had a dictionary size of 6,828,514 and were used for all experiments.

4.2 Model and Hyper-parameters

Figure 2 presents the architecture of the NER model proposed by Yadav et al. (2018). The model features Character Bi-LSTM, Word features, Word Bi-LSTM, and Word CRF layer for generating BIO tags for the named entities. The above model was used as is with minor modifications in hyper-parameters. The word embeddings size of 200-d was used, train_embeddings was set to false, and batch_size was set to 25. All other parameters were set to the default values given in the model proposed by Yadav et al. (2018).

4.3 NER datasets

The WEAVE corpus of the present study was randomly split into training, development, and test set with 60 documents in each set. The official training, development, and test set of CHEMDNER-patents task (Krallinger et al., 2015c) was used without modification.

4.4 Preprocessing

The WEAVE corpus in the BRAT standoff format was converted into CoNLL 2003 BIO format and truncated to the Examples section. The resulting WEAVE corpus had 73,522 sentences, 3,453,525 tokens, and 15,782 unique tokens. The CHEMDNER-patents corpus in a tab-separated format was converted into CoNLL 2003 BIO format before being used in training and evaluation of the model. The resulting CHEMDNER-patents corpus had 73,383 sentences, 2,511,006 tokens, and 51,570 unique tokens.

5 Analysis

To better understand the WEAVE corpus's baseline performance, we conducted several experiments involving BC-V corpus and its combinations with the WEAVE corpus. In Tables 8 and 9 we present the results of experiments on various combinations of WEAVE and BC-V datasets.

Based on the simple NER model (Yadav et al., 2018), the best result in terms of macro-averaged F-measure was the model on standalone WEAVE corpus and tested on WEAVE test set with 91.37%. Followed by a model trained on BC-V + WEAVE corpus and tested on the WEAVE test set with 91.34%. In comparison, the top-performing team in the BC-V CHEMDNER-patents task had an F-score of 89.37% (Krallinger et al., 2015c). Whereas the model trained on standalone BC-V corpus and tested on BC-V test corpus had an F-measure of 80.89%. The model's worst performance was when trained on WEAVE corpus and tested on the BC-V test set; the F-measure was 29.93%.

The results validate the linguistic structure of the title and abstract of a patent is very different from that of the Examples section. Hence, when combined with the CHEMDNER-patents corpus, the WEAVE corpus are complementary; without losing precision, we have an increase in the recall of the NER model. This also supports our assertion of the need for a focused dataset covering the Examples section of patents. The combined corpus can perform very close to the state-of-the-art results in chemical NER. This combination also gives us many high-quality annotations 199,763 (100,129 WEAVE + 99,634 BC-V) to develop better chemical NER models. The IAA value of 98.73% on 45 documents subset and the best NER model with F-measure of 91.37% is instructive of the NER model's simple nature. There is good scope for researching better NER models, which can reduce this difference.

Training	Development	Test	Precision	Recall	F1
BC-V	BC-V	BC-V	78.62	83.30	80.89
BC-V	WEAVE	BC-V	78.21	80.21	79.19
WEAVE	BC-V	BC-V	35.68	25.78	29.93
WEAVE	WEAVE	BC-V	32.40	24.65	27.99
BC-V + WEAVE	BC-V	BC-V	74.50	78.77	76.58
BC-V + WEAVE	WEAVE	BC-V	73.33	76.34	74.80
BC-V + WEAVE	BC-V + WEAVE	BC-V	73.84	77.93	75.83

Table 8: Experimental results with BC-V Test corpus

Training	Development	Test	Precision	Recall	F1
BC-V	BC-V	WEAVE	67.08	50.80	57.82
BC-V	WEAVE	WEAVE	73.32	48.38	58.29
WEAVE	BC-V	WEAVE	93.24	89.11	91.13
WEAVE	**WEAVE**	**WEAVE**	**93.55**	**89.29**	**91.37**
BC-V + WEAVE	BC-V	WEAVE	92.91	88.76	90.79
BC-V + WEAVE	WEAVE	WEAVE	92.54	88.74	90.60
BC-V + WEAVE	**BC-V + WEAVE**	**WEAVE**	**93.43**	**89.34**	**91.34**

Table 9: Experimental results with WEAVE Test corpus.

6 Discussion

Our results show that a focused annotated NER dataset with a simple NER model can achieve near state-of-the-art results. Complementary datasets can achieve high recall without sacrificing the precision of the chemical NER model. This is illustrated by the rows highlighted as bold in Table 9. The reuse of the existing manually annotated dataset results in substantial savings in manual annotation effort.

Chemical NER models with high precision and recall can be used for downstream processing and analysis of chemical reactions in patents. The present annotated dataset would help better temporal modeling of the synthetic procedures given in the Examples section of patents.

We propose to explore more complex NER models. These models can better account for the high IAA values reported by us. In the future, we would explore the possibility of extending this dataset to chemical reaction role labeling for the identified chemical entities.

7 Supporting Information

The WEAVE corpus described in this paper is available at Github repository: https://github.com/nv-ravindra/the-weave-corpus

Acknowledgments

We thank Vincatis Technologies Private Limited, Hyderabad and anonymous reviewers for their help with this publication.

References

Saber A Akhondi, Alexander G Klenner, Christian Tyrchan, Anil K Manchala, Kiran Boppana, Daniel Lowe, Marc Zimmermann, Sarma ARP Jagarlapudi, Roger Sayle, Jan A Kors, et al. 2014. Annotated Chemical Patent Corpus: A Gold Standard for Text Mining. *PloS One*, 9(9):e107477.

Michael Bada, Miriam Eckert, Donald Evans, Kristin Garcia, Krista Shipley, Dmitry Sitnikov, William A Baumgartner, K Bretonnel Cohen, Karin Verspoor, Judith A Blake, et al. 2012. Concept annotation in the CRAFT corpus. *BMC Bioinformatics*, 13(1):161.

Leonardo Campillos, Louise Deléger, Cyril Grouin, Thierry Hamon, Anne-Laure Ligozat, and Aurélie Névéol. 2018. A French clinical corpus with comprehensive semantic annotations: development of the Medical Entity and Relation LIMSI Annotated Text corpus (MERLOT). *Language Resources and Evaluation*, 52(2):571–601.

Connor W Coley, Regina Barzilay, Tommi S Jaakkola, William H Green, and Klavs F Jensen. 2017. Prediction of Organic Reaction Outcomes using Machine Learning. *ACS Central Science*, 3(5):434–443.

Peter Corbett, Colin Batchelor, and Simone Teufel. 2007. Annotation of Chemical Named Entities. In *Biological, translational, and clinical language processing*, pages 57–64, Prague, Czech Republic. Association for Computational Linguistics.

Tiago Grego, Piotr Pezik, Francisco M Couto, and Dietrich Rebholz-Schuhmann. 2009. Identification of Chemical Entities in Patent Documents. In *International Work-Conference on Artificial Neural Networks*, pages 942–949. Springer.

Cyril Grouin and Aurélie Névéol. 2014. De-identification of clinical notes in French: towards a protocol for reference corpus development. *Journal of Biomedical Informatics*, 50:151 – 161. Special Issue on Informatics Methods in Medical Privacy.

Ming-Siang Huang, Po-Ting Lai, Pei-Yen Lin, Yu-Ting You, Richard Tzong-Han Tsai, and Wen-Lian Hsu. 2020. Biomedical named entity recognition and linking datasets: survey and our recent development. *Briefings in Bioinformatics*. Bbaa054.

Corinna Kolarik, Roman Klinger, Christoph M. Friedrich, Martin Hofmann-Apitius, and Juliane Fluck. 2008. Chemical Names: Terminological Resources and Corpora Annotation. In *Workshop on Building and evaluating resources for biomedical text mining (6th edition of the Language Resources and Evaluation Conference)*, pages 51–58.

Martin Krallinger, Florian Leitner, Obdulia Rabal, Miguel Vazquez, Julen Oyarzabal, and Alfonso Valencia. 2015a. CHEMDNER: The drugs and chemical names extraction challenge. *Journal of Cheminformatics*, 7(S1):S1.

Martin Krallinger, Obdulia Rabal, Florian Leitner, Miguel Vazquez, David Salgado, Zhiyong Lu, Robert Leaman, Yanan Lu, Donghong Ji, Daniel M Lowe, et al. 2015b. The CHEMDNER corpus of chemicals and drugs and its annotation principles. *Journal of Cheminformatics*, 7(1):1–17.

Martin Krallinger, Obdulia Rabal, Analia Lourenço, Martin Perez Perez, Gael Perez Rodriguez, Miguel Vazquez, Florian Leitner, Julen Oyarzabal, and Alfonso Valencia. 2015c. Overview of the CHEMDNER patents task. In *Proceedings of the fifth BioCreative challenge evaluation workshop*, pages 63–75.

Sheshera Mysore, Zach Jensen, Edward Kim, Kevin Huang, Haw-Shiuan Chang, Emma Strubell, Jeffrey Flanigan, Andrew McCallum, and Elsa Olivetti. 2019. The Materials Science Procedural Text Corpus: Annotating Materials Synthesis Procedures with Shallow Semantic Structures. In *Proceedings of the 13th Linguistic Annotation Workshop*, Florence, Italy. Association for Computational Linguistics.

Mariana Neves and Ulf Leser. 2012. A survey on annotation tools for the biomedical literature. *Briefings in Bioinformatics*, 15(2):327–340.

Dat Quoc Nguyen, Zenan Zhai, Hiyori Yoshikawa, Biaoyan Fang, Christian Druckenbrodt, Camilo Thorne, Ralph Hoessel, Saber A. Akhondi, Trevor Cohn, Timothy Baldwin, and Karin Verspoor. 2020. ChEMU: Named Entity Recognition and Event Extraction of Chemical Reactions from Patents. In *Advances in Information Retrieval*, pages 572–579, Cham. Springer International Publishing.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Marwin HS Segler, Mike Preuss, and Mark P Waller. 2018. Planning chemical syntheses with deep neural networks and symbolic AI. *Nature*, 555(7698):604–610.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. BRAT: a Web-based Tool for NLP-Assisted Text Annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Vikas Yadav and Steven Bethard. 2018. A Survey on Recent Advances in Named Entity Recognition from Deep Learning models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2145–2158, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Vikas Yadav, Rebecca Sharp, and Steven Bethard. 2018. Deep Affix Features Improve Neural Named Entity Recognizers. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 167–172, New Orleans, Louisiana. Association for Computational Linguistics.

Increasing accuracy of a semantic word labelling tool based on a small lexicon

Hugo Sanjurjo-González
University of Deusto, Spain
hugo.sanjurjo@deusto.es

Abstract

Semantic annotation has become an important piece of information within corpus linguistics. This information is usually included for every lexical unit of the corpus providing a more exhaustive analysis of language. There are some resources such as lexicons or ontologies that allow this type of annotation. However, expanding these resources is a time-consuming task. This paper describes a simple NLP baseline for increasing accuracy of the existing semantic resources of the UCREL Semantic Analysis System (USAS). In our experiments, Spanish token accuracy is improved by up to 30% using this method.

1 Introduction

Apart from raw texts, a corpus can include extra linguistic information by way of annotation. Most common types of annotation are grammatical, semantic, prosodic and historical. The semantic one has become an important piece of information within the corpus linguistics research field. A corpus with this information is a useful resource to extract knowledge from a real context: as Navarro et al. (2005) state, it can be considered as a semi-structured database that offers deep information about human knowledge, concepts and relations among them.

Semantic annotation in corpus linguistics tends to recognise semantic categories and concepts at different syntactic levels, such as word level, phrase level or sentence level (Piao et al., 2018). For this purpose, the information about grammatical tags and NER (Named-Entity Recognition) classes contribute to determine lexical semantics to some extent, but they are not sufficiently informative (Abzianidze and Bos, 2017). Semantic annotation tries to overcome these barriers by adding new categories.

In this paper, we describe an NLP baseline that increases accuracy of a semantic role labelling tool that makes use of a small semantic lexicon in Spanish language (Piao et al., 2015) based on the USAS tagset (Archer et al., 2002). We are able to increase accuracy by means of a very simple strategy that makes use of freely-available NLP toolkits such as NLTK (Bird et al., 2009) and Spacy (Honnibal and Montani, 2017). A novel approach using WordNet similarity based on the information content theory (Resnik, 1995) is also employed in order to search synonyms of unknown words and, therefore, increase lexical accuracy. As a proof of concept, we carried out different experiments with texts from the finance domain.

The article is organised as follows: Section 2 explains our approach together with the different processes that are executed. Implementation is described in Section 3. We show different experiments in Section 4. Last, Section 5 outlines conclusions and future directions.

2 Overview of our approach

The USAS lexicon is based on the Longman Lexicon of Contemporary English (McArthur, 1986), which ensures, up to a certain point, the linguistic validity and motivation of this resource. There are 21 major discourse fields, expanding into 232 category labels[1]. USAS employs a group of labels in an attempt to include most meanings of the lexicalised unit. To employ USAS lexicon, we need to extract lemma and grammatical annotation for each word. Table 1 shows an example of entry for the word *business*.

Regarding the Spanish version, it contains around 10,000 words and 5,000 multiword expressions, and most of them are Spanish named entities such as places or locations. As a consequence of its

[1]More information can be found in (Archer et al., 2002)

Proceedings of the 17th International Conference on Natural Language Processing, pages 10–14
Patna, India, December 18 - 21, 2020.

Table 1: Example of lexicon entry.

Lemma	POS	Semantic tags
business	noun	I2.1 A1.1.1 A5.1+++

reduced size, accuracy of the Spanish USAS lexicon is limited if it is used in specific text domains. For instance, only 3.30% of the lexicon entries belong to the finance domain. If this is the only resource employed for tagging, there will be many words that will not have any tag, and thus, many words will be incorrectly tagged as unknown because they do not appear in the lexicon (e.g. *índice* - index).

A more in-depth analysis reveals that some of these words are lemmatised incorrectly (e.g. *véase* (note) is lemmatised as *véase* instead of *ver*). In addition, some words appear in the lexicon with only one grammatical category when they can belong to different categories (e.g. *mucho* (many/much) can be an adverb and adjective in Spanish).

To solve these problems, the simplest solution is to add new entries to the lexicon, however this is a very time-consuming task. Another solution is trying to improve results of the operations required by USAS such as lemmatisation or grammatical annotation. We can also try to incorporate other techniques such as stemming in order to match the stem of the word with another stem in the lexicon. In order to achieve that, we can simply employ available Spanish resources from NLP toolkits such as Spacy and NLTK. In the rest of this section, we will describe how lexicon accuracy may be increased using some preprocessing techniques in a specific domain such as the finance one.

2.1 Analysing finance domain texts

In this stage, we built a corpus of texts from the finance domain in order to analyse its main features such as most frequent words, keywords, collocations etc. More concretely we selected the Annual Report of the Banco de España (1998-2019) (BDE, 2020) with the exception of the 2013 edition, since it was used for validation purposes. These documents review economic and financial developments in the Spanish economy and are composed of 19 samples and 2,841,826 words.

Analysis of this corpus reveals that this type of texts appear to have many acronyms such as PIB (*Producto Interior Bruto* - Gross Domestic Product), numbers with different formats (2005, 36,3%, 540.000, 3,25), currency symbols, proper names (Miguel Fernández Ordóñez Antonio Rosas), geographical names (Torrellano-Elche, *Eurozona*), organisations names (*Banco de España*) and words in languages different from Spanish (financial institutions) as well as other jargon of this domain.

2.2 Lemmatisation

As we previously mentioned, USAS lexicon entries are composed of lemma, POS tag and semantic tag (see Table 1). Thus, it is necessary to include a Spanish lemmatiser. NLTK does not offer this tool for Spanish language, and Spacy includes one but it has some errors. For instance: *reclamaciones* as *reclamaciones* (claims in English) or *como* (like) as *comer* (eat), para (for) as *parir* (give birth), among others.

Using full words instead of lemmas also entails some errors because most of them do not appear in the lexicon. For instance, *músculos* - muscles instead of *músculo* - muscle. For this reason, we make use of the NLTK stemmer, which returns words' bases or roots.

2.3 Grammatical annotation

We also need to annotate each word grammatically. English USAS employs CLAWS (Garside, 1987), a highly sophisticated grammatical tagger. However, there is not an equivalent for other languages, so in this case USAS for Spanish employs a simplified version of the grammatical tagset that includes the basic grammatical elements.

For this purpose, we employ Spacy grammatical tagger since it offers a relatively adequate performance. Nevertheless, some words are incorrectly tagged, mainly some nouns or even adjectives that were tagged as proper nouns because of their initial uppercase. For instance: *Informe* (Report) and *Anual* (Annual). As a consequence, the semantic tagger would return no tag for all these words. To overcome this problem we search for words without grammatical tags in the lexicon at the end of the process, that is, as a final measure to return semantic tag candidates.

2.4 Identifying named entities and foreign words

We make use of a corpus of names included in NLTK (Kantrowitz, 2020), for instance Alberto. This allows us to identify any name in different languages. We also employ some gazetteers for

geographical locations (e.g. Madrid) and the previously mentioned corpus for identifying English words that usually appear in financial texts (e.g. Exchange).

Including a NER tagger for Spanish is also an option, however according to our experiments this tool recognises many foreign words such as organisations or even locations (e.g. Cash). For this reason, if we included it, it would return many false positives.

2.5 Identifying other elements

In order to identify any format number, mathematical operations and symbols as well as some other elements like abbreviations, we formulate patterns using Perl compatible regular expressions.

2.6 Computing WordNet synonyms

We also wanted to make use of a novel approach for identifying unknown words that were not tagged in previous steps. To do that, we try to get synonyms of the unknown words, since synonyms often have the same semantic function.

We employ sense similarity of the information content of the corpus compiled at the first stage of this approach. Our premise is that if one word is missing from the lexicon there are many possibilities that this word has a synonym in the previously compiled corpus.

We create an information content dictionary of the corpus in order to employ similarity based on the WordNet synsets. To measure similarity we employ Lin measure (Lin, 1998).

3 Implementation and deployment

We develop all the components of the tagger following the specifications proposed in the previous section. Fig. 1 shows a simplified workflow of this process that can be described as follows:

1. First, the tagger searches if the lemma of the word together with its grammatical tag is in the lexicon. If it is, we already have the tag for the word.

2. If it is not, the tagger searches if the word with its grammatical tag is in the lexicon.

3. If not, we employ the stemmed version of the word and the lexicon together with the grammatical tag.

Table 2: Evaluation of accuracy.

Sample text size	Correct	Partially correct
13,331 words	86.26%	2.21%
7,064 words	86.71%	1.33%

4. If we do not have any results, we try the same without using grammatical annotation as a consequence of the possible errors of the grammatical tagger.

5. After that, words without semantic tags are analysed in order to identify named entities and foreign words.

6. Subsequently, regular expressions are used to match numbers and abbreviations, among others.

7. For the rest of the unmatched words, the tagger will search a synonym of the word using the information content and its similarity based on WordNet synsets. We get a list of candidates according to their similarity index and search them in the lexicon.

8. If similar words cannot be calculated with that word or its lemma, it will be set as a semantic tag 'Z99' or unknown word.

4 Experiments

In the absence of resources for validating our tool we needed to build a custom-made gold standard. This is a consequence of the USAS tagset, a very specific classification system, and the Spanish language, which has less lexical coverage than English language using this resource. We extracted some sample texts that were not included in the corpus together with some texts from independent sources. The size of the gold standard is 20,395 words. This size is a consequence of the laborious task of manual annotation.

In order to evaluate the accuracy, we followed the same metrics as (Piao et al., 2015). A first metric refers to those instances where the first candidate tag is correct, and a second metric makes reference to the cases where the other tags in the list are correct or closely related to the word sense. These results are shown in Table 2

As we can see in Fig. 2, results have been improved around 30% in comparison with using only Lemma – POS method.

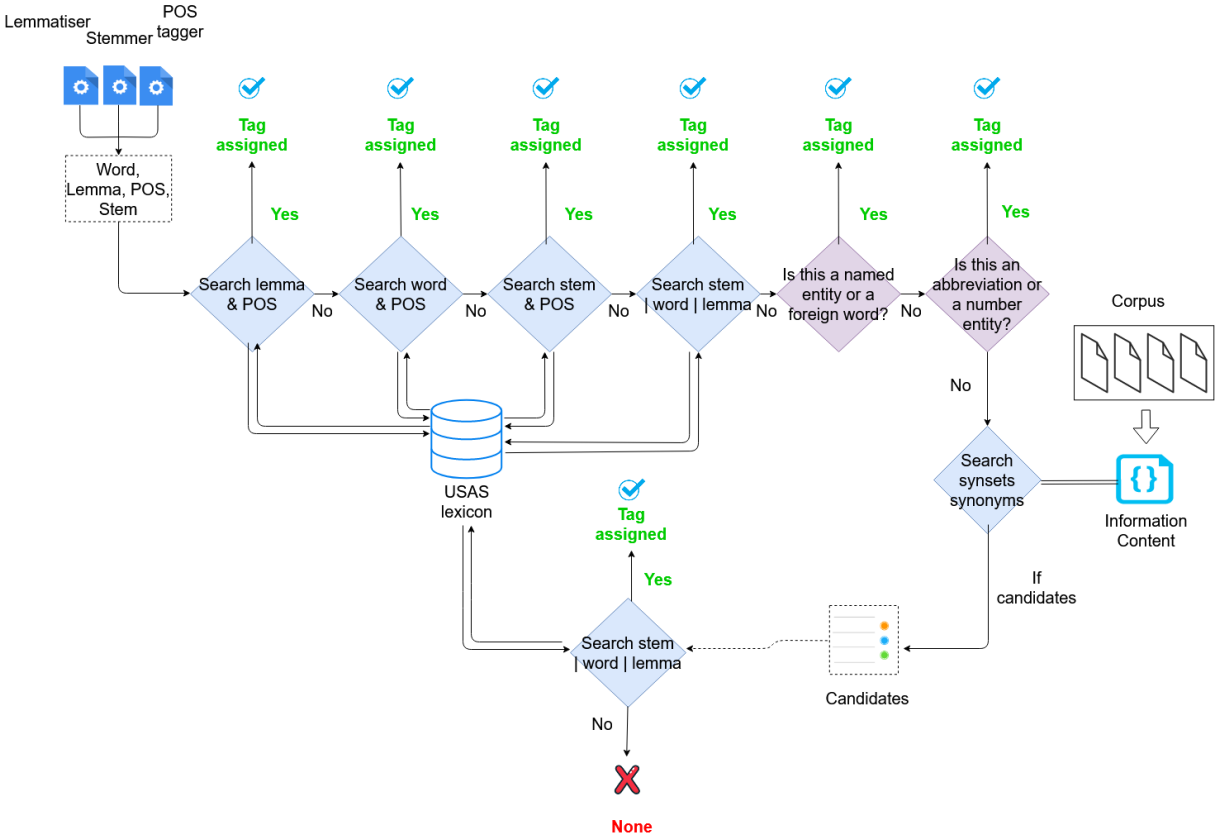


Figure 1: Simplified workflow diagram of the tagger.

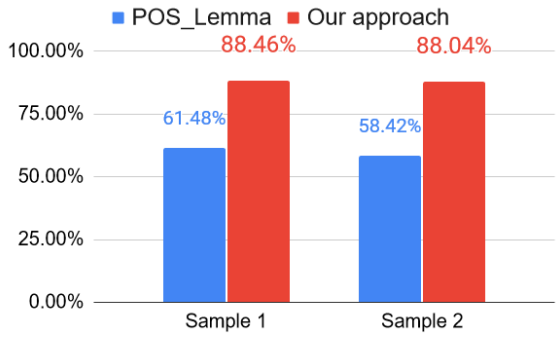


Figure 2: Experiment results.

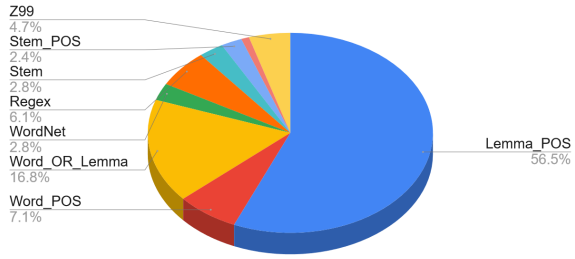


Figure 3: % of words that were tagged for each subprocess.

Fig. 3 shows percentages of words that were tagged for each subprocess of the tagger. As it can be seen, the proposed baseline tags about 44% of the words. WordNet synonym method did not return any significant results, maybe as a consequence of the absence of a basis of finance elements in the lexicon. Its inclusion only improves accuracy around 0.15% according to our experiments, so it is not significant.

Last, the confusion matrix of the semantic tagger according to the 21 major discourse fields of the USAS taxonomy can be found in the Supplementary Material. Confusion matrix of all the 232 subcategories would be a more detailed option but the representation of all the subcategories may be slightly confusing. The most remarkable issues are the following:

- Many words from the rest of the categories are incorrectly tagged using Z category. More specifically with 'Z99' tag. That means that there are many words that our tagger cannot recognised and as a consequence they are tagged as unknown.
- Another issue is related to the words belonging to the N category (Numbers and Measure-

ment) that are wrongly tagged using the A category (General and Abstract terms). Words such as *índice* (index) or *tasa* (rate, fee) are not correctly identified.

- Last, it should be mentioned that words belonging to A category are tagged incorrectly using the rest of the categories. One explanation may be the own definition of this category, general and abstract terms.

5 Conclusions and further work

The main contribution of this paper is a strategy that utilises existing NLP toolkits such as NLTK and Spacy to preprocess texts in order to obtain better results using only a small lexicon as source of semantic information. This strategy is implemented following a simple and straightforward approach. Empirical results are reported and compared across an ad hoc gold standard based on texts from the finance domain.

This study also introduced a novel approach for extending lexical accuracy of semantic lexicons by means of synsets similarity of WordNet which did not provide the expected results, maybe as a consequence of the limited lexicon.

We hope that this approach could be easily extended to other domains and even with under-resourced languages. Therefore, expected future work includes reproducing the good results obtained in other text domains and employing languages different from Spanish or English. We also need to investigate how to take advantage of the semantic similarity provided by WordNet or even word embeddings using taxonomies like USAS.

References

Lasha Abzianidze and Johan Bos. 2017. Towards universal semantic tagging. In *IWCS 2017—12th International Conference on Computational Semantics—Short papers*.

Dawn Archer, Andrew Wilson, and Paul Rayson. 2002. Introduction to the usas category system. retrieved from ucrel semantic analysis system (usas) website. http://ucrel.lancs.ac.uk/usas/usas_guide.pdf. Accessed: 2020-09-16.

BDE. 2020. Banco de españa - publicaciones anuales website. https://www.bde.es/bde/es/secciones/informes/Publicaciones_an/Informe_anual/. Accessed: 2020-09-16.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Roger Garside. 1987. The claws word-tagging system. *The Computational analysis of English: A corpus-based approach. London: Longman*, pages 30–41.

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*, 7(1).

Mark Kantrowitz. 2020. Mark kantrowitz corpora names website. http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/nlp/corpora/names/. Accessed: 2020-09-16.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Machine Learningˆ* Proceedings of the Fifteenth International Conference (ICML'98)*, pages 296–304.

Tom McArthur. 1986. *Longman lexicon of contemporary English*. Longman.

Borja Navarro, Patricio Martínez-Barco, and Manuel Palomar. 2005. Semantic annotation of a natural language corpus for knowledge extraction. In *International Conference on Application of Natural Language to Information Systems*, pages 365–368. Springer.

Scott SL Piao, Francesca Bianchi, Carmen Dayrell, Angela D'egidio, and Paul Rayson. 2015. Development of the multilingual semantic annotation system. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1268–1274.

Scott SL Piao, Paul Rayson, Dawn Knight, and Gareth Watkins. 2018. Towards a welsh semantic annotation system. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

P Resnik. 1995. Using information content to evaluate semantic similarity. In *Proc. 14th International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, Canada*, pages 448–453.

Treatment of optional forms in Mathematical modelling of Pāṇini

Anupriya Aggarwal, Malhar Kulkarni
Indian Institute of Technology Bombay, Mumbai
anupriya@iitb.ac.in, malharku@hss.iitb.ac.in

Abstract

Pāṇini in his *Aṣṭādhyāyī* has written the grammar of Sanskrit in an extremely concise manner in the form of about 4000 *sūtras*. We have attempted to mathematically remodel the data produced by these *sūtras*. The mathematical modelling is a way to show that the *Pāṇinian* approach is a minimal method of capturing the grammatical data for Sanskrit which is a natural language. The *sūtras* written by *Pāṇini* can be written as functions, that is for a single input the function produces a single output of the form y=f(x), where x and y is the input and output respectively. However, we observe that for some input *dhātus*, we get multiple outputs. For such cases, we have written multivalued functions that is the functions which give two or more outputs for a single input. In other words, multivalued function is a way to represent optional output forms which are expressed in *Pāṇinian* grammar with the help of 3 terms i.e. *vā*, *vibhaṣā*, and *anyatarasyām*. Comparison between the techniques employed by *Pāṇini* and our notation of functions helps us understand how *Pāṇinian* techniques ensure brevity and terseness, hence illustrating that *Pāṇinian* grammar is minimal.

1 Introduction

Pāṇini's Aṣṭādhyāyī is 'almost an exhaustive grammar for any human language with meticulous details yet small enough to memorize it' (Kulkarni, 2016). Such an exhaustive grammar is ideal to be used for artificial language processing. Briggs (Briggs, 1985) even demonstrated in his article the salient feature of Sanskrit language that can make it serve as an artificial language. Although, this is not a new concept, various efforts in mathematical modelling of Indian languages have been done before. Joseph Kallrath in his book 'Modeling Languages in Mathematical Optimization' says that 'a modeling language serves the need to pass data and a mathematical model description to a solver in the same way that people especially mathematicians describe those problems to each other' (Kallrath, 2013). Mathematical modelling of languages also impacts our understanding of the language and its grammar. As scholars are delving into the question of formalizing various natural languages, it is also having an impact on how we understand the language itself. Recent work in theoretical and computational linguistics has influenced the interpretation of grammar (Scharf, 2008). We have followed a similar approach, wherein we have modelled the *Pratyayas* in Sanskrit in the form of functions with the help of *Pāṇinian sūtras*.

Similar to mathematical functions which can be expressed as f(x)=y where x is the input and y is the output of function f; 'the *sūtras* too look for their preconditions in an input environment. The effects produced by *sūtras* become part of an ever-evolving environment which may trigger other' (Sohoni & Kulkarni, 2018). For the grammar to fit mathematical functions, we 'need a strong and unambiguous grammar which is provided by *Maharishi Pāṇini* in the form of *Aṣṭādhyāyī*' (Agrawal, 2013).

Statistical analysis of a language is a vital part of natural language processing (Goyal, 2011). According to how components of the target linguistic phenomenon are realized

Proceedings of the 17th International Conference on Natural Language Processing, pages 15–21
Patna, India, December 18 - 21, 2020.

mathematically, available models of language evolution can be classified as rule-based and equation-based models. Equation-based models tend to transform linguistic and relevant behaviors into mathematical equations (Tao Gong, 2013), which is what we have attempted in this paper.

Ambiguity is inherent in the Natural Language sentences (Tapaswi & Jain, 2012), and hence Sanskrit being a natural language also has certain ambiguities. The ambiguity that we are dealing with in this paper is that a single *dhātu* combined with a single pratyaya can result in two or more optional forms. Mathematical modelling of such natural languages can help to remove this ambiguity. Traditionally too, there have been attempts by various scholars like *Kātyāyana*, *Patanjali* and *Bhartṛhari* to provide extensive commentaries which contain explanations for various aspects of the grammar. They do not question *Pāṇini's* basic model, but rather explain it, refine it and complete it (Huet, 2003). Explanations and clarifications in the form of various *vārtikas* also come handy while dealing with ambiguities. However, here we are diverging from the traditional approach and writing functions in order to model the grammatical data.

To account for more than two forms of a word, *Pāṇini* uses optional form rules to state that alternate forms are also possible. For example, *sūtra* (rule) 1.2.3 *vibhaṣorṇoḥ* states that 'After the verb *ūrṇa* 'to cover', the affix beginning with the augment iṭ is regarded optionally like ṅit (Source, 2020)'. We have used multivalued functions to denote such optional forms in our system of representing the *pratyayas* as functions.

2 Methodology

We are here attempting to mathematically model the data produced by the *sūtras* for which we started with compiling the list of *dhātus* and their respective derived *dhātus* with different *pratyayas* like from the Kridantkosh of Pushpa Dikhshita Vol.1 (Dikshita, 2014), sanskritworld.in (Dhaval Patel, n.d.), Siddhananta Kaumudi of Bhattoji Dikshita (S.C.Vasu, 1905), The Madhaviya DhātuVritti (Sayanacarya, 1964) and the roots, verb-forms and primary derivatives of the Sanskrit Language by W.D.Whitney (Whitney, 1885). The list of *dhātus* without the application of any *pratyaya* are considered as x, after the application of the concept of *anubandhas*. *Anubandhas* have a very prominent role to play in the *Pāṇinian* system of Sanskrit grammar. It literally means 'what is attached to'. It has been used by all ancient authorities on Sanskrit grammar who have come after *Pāṇini*, right from *Kātyāyana* to *Nageśa*. However, *Pāṇini* has used the term '*it*' to describe the *anubandhas*. M. Williams dictionary (Williams, 2008 revised) defines *anubandhas* as an indicatory letter or syllable attached to roots etc., marking some peculiarity in their inflection e.g. an '*i*' attached to roots denotes the insertion of a nasal before their final consonant. According to *Nyāyakosa*, *anubandha* is a letter that is attached to the stem (*prakṛti*), termination (*pratyaya*), augment (*āgama*) or a substitute (*ādesha*) to indicate the occurrence of some special modifications such as *vikaraṇa*, *āgama*, *guṇa* or *vṛddhi*, accent etc. But it is dropped from the finished word i.e. *pada*. The use of *anubandha* is one of the crucial steps *Pāṇini* has taken to ensure the brevity and terseness of his work. We can say that *anubandhas* do form part of the *pratyayas* etc. to which they are found appended (Devasthali, 1967). But before we directly start writing our functions, we need to define the input set which comprises of *dhātus* from the *Dhātupatha* as well as the derived *dhātus* without *anubandhas*.

Let A be a set of all the *dhātus* after the *anubandhas* have been removed. These primary *dhātus are* 1943 in total. However, the input *dhātus* are not limited to these *dhātus* in set A. We can also derive a new *dhātu* set B by adding a *san pratyaya* to the *dhātus of* set A. The items in set B can be called *dhātus by* following the grammatical rule laid down by *Pāṇini,* '3.1.32 *sanādyantāḥ dhātavaḥ*' which says that 'all roots ending with

***Sūtra* numbers**	***Pratyaya***
3.1.5	*san*
3.1.8	*kyac*
3.1.9	*kāmyac*
3.1.11	*kyaṅ*
3.1.13	*kyaṣ*
3.1.20	*ṇiṅ*
3.1.21	*ṇic*
3.1.22	*yaṅ*
3.1.27	*yak*
3.1.28	*āy*
3.1.29	*īyaṅ*

Table 1: List of *San pratyayas* in *Aṣhṭādhyāyī* with their respective *sūtra* numbers

the *pratyayas* starting with san are called *dhātu.*

Hence the input x is defined as,

$$x \in (A \cup B)$$

In this paper we will focus on the multivalued functions that give two or more outputs for the same input *dhātu* of the form $f(x)=\begin{cases} y_1 \\ y_2 \end{cases}$ if there are two optional forms; $f(x)=\begin{cases} y_1 \\ y_2 \\ y_3 \end{cases}$ if there are three optional forms and so on.

3 Notation

Let x be the input *dhātu*. For the purpose of writing these functions, we start enumerating the syllables from left to right or from right to left depending upon that particular function. We can denote x as, x= (….,x(2),x(1))= (x'(1),x'(2),….). x can be a consonant (C) or a vowel (V) and they are denoted by

C'(i)= i^{th} consonant from left;
V'(i)= i^{th} vowel from left;
C(i)= i^{th} consonant from right,
V(i)= i^{th} vowel from right.

cura =	*c*	*u*	*r*	*a*
Right to left	x(4)	x(3)	x(2)	x(1)
Left to right	x'(1)	x'(2)	x'(3)	x'(4)

Table 2: The numbers 1, 2, 3,… signify the position of the syllable. The notation x (unprimed) is used when the syllables are counted right to left, and the notation x' is used when the syllables are counted left to right.

For example: If x = *cura*, then

Conversion are denoted by a right arrow with a number on the top. The number denotes the location of the conversion.

For example, $x[a \xrightarrow{2} \bar{a}]$ denotes that in the *dhātu* x, *a* which is at the 2nd place from the right is getting replaced with *ā*.

We also define a '+ operator' to explain the change of syllables when two syllables combine. In Sanskrit language when two syllables come closer, for the ease of pronunciation (in most cases) it gets replaced by another syllable or a combination of syllables. For example: *ū+i=vi, e+i=ayi, o+i=avi, d+ta=tta, ch+t= ṣṭa, j+ta=kta, dh+ta=dhda, bh+ta=bdha, h+ta= ṇḍha*. Note that although the '+ operator' may look similar to the concept of *Sandhi* in Sanskrit, it is totally based on our need to fit our dataset and does not encompass the broad concept of *Sandhi*.

4 A function p(x)

This function is not a *pratyaya* function, but it is required to write the *pratyaya* function. Thus, it would be helpful to define it here. The *dhātus* which have two or more vowels are called *udātta*, and when a suffix is added to them an additional '*i*' comes. Such *dhātus* are called *seṭ* (literally meaning 'with *iṭ*'). For *dhātus which* have one vowel, we need to see the instructions given in the *Dhātupaṭha*. They can either be *seṭ* or *aniṭ* depending upon the given instructions given. Example of one such instruction is '*bhu sattayām*| *'udāttḥ parasmaibhāṣḥ*'| It says that '*i*' will come as the *prayogsamavāyī svara* is *udāttḥ*.
The function p(x) is defined by,

१ भू सत्तायाम् । 'उदात्तः परस्मै-
भाषः' ॥
अथ षट्त्रिंशत्तवर्गीयान्ता
आत्मनेपदिनः ।
२ एध वृद्धौ ।
३ स्पर्ध सङ्घर्षे ।
४ गाधृ प्रतिष्ठालिप्सयोर्ग्रन्थे च ।
५ बाधृ लोडने ।
६ नाथृ-
७ नाधृ याच्ञोपतापैश्वर्या-
शीःषु ।

Figure 1: Example of an Instruction given in the *Dhātupatha*.

$$p(x) = \begin{cases} \text{i if x is } set, \\ \text{0 if x is } aniṭ. \end{cases}$$

5 Multivalued functions

The words used for optionality by *Pāṇini* are vā, *vibhaṣā*, *anyatarasyām*. *vā* appears 136 times, *vibhaṣā* appears 258 times and, *anyatarasyām* appears 161 times respectively in *Aṣṭādhyāyī*; including the ones that occur in *Anuvritti*[1]. *Pāṇini* and all the commentators have given us no indication that they are supposed to be anything but synonyms. But the modern scholar Paul Kiparsky has wondered how could this be so, because *Pāṇini* has vowed to eliminate every needless extraneous syllable and there must be a deeper reason to suggest the use of three different terms. Hence, he has propounded the hypothesis in his well-argued study *Pāṇini* as a 'variationist' that the three terms *vā, vibhaṣā, anyatarasyām* refer respectively to three different kinds of options: those that are preferable (*vā*), those that

Word	Occurrence	Usage
vā	136 times	preferable
vibhaṣā	258 times	marginal
anyatarasyām	161 times	simple options

Table 3: Words used for optionality by *Pāṇini*

are marginal (*vibhaṣā*) and those that are simple options(*anyatarasyām*) (Sharma, 2018).

One such case which results in such optional forms is represented in the table below where the addition and absence of '*i*' results in two forms and the change of '*h'* syllable to two different syllables further results in two forms. Thus, we end up with three forms of the same word.

Let us look at an example for this case for x = *muh*:

$$\text{tum(muh)} = \begin{cases} \text{x}\left[i\ u \xrightarrow{2} e\ o\right] + 0 + tum \\ \text{x}\left[i\ u \xrightarrow{2} e\ o\right] + 0 + ḍhum \\ \text{x}\left[i\ u \xrightarrow{2} e\ o\right] + i + tum \end{cases} = \begin{cases} mogdhum \\ moḍhum \\ mohitum \end{cases}$$

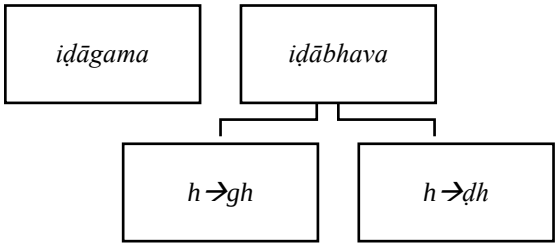


Figure 2:Multivalued functions

6 Cases for multivalued functions

Some cases for multivalued functions are displayed below[2].

Some Multivalued functions for *Tumun*

Pratyaya

Case I:

If x ϵ {*svṛ sū dhū*}, then

$$\text{tum(x)} = \begin{cases} \text{x}\left[i/\text{īu}/\text{ūṛ}/\text{ṝ} \xrightarrow{1} e\ o\ ar\right] + i + tum \\ \text{x}\left[i/\text{īu}/\text{ūṛ}/\text{ṝ} \xrightarrow{1} e\ o\ ar\right] + 0 + tum \end{cases}$$

x	$\begin{cases} x\left[i/\text{īu}/\text{ūṛ}/\text{ṝ} \xrightarrow{1} e\ o\ ar\right] + i \\ x\left[i/\text{īu}/\text{ūṛ}/\text{ṝ} \xrightarrow{1} e\ o\ ar\right] + 0 \end{cases}$	tum(x)
sū	*svi* *so*	*svitum* *sotum*
svṛ	*svari* *svar*	*svaritum* *svartum*

Case II:

If x has two syllables such that x(1)= *ṛ*, then

$$\text{tum(x)} = \begin{cases} x\left[\text{ṛ} \xrightarrow{1} ar\right] + i + tum \\ x\left[\text{ṛ} \xrightarrow{1} ar\right] + \text{ī} + tum \end{cases}$$

x	$\begin{cases} \text{x}\left[\text{ṛ} \xrightarrow{1} \text{ar}\right] + \text{i} \\ \text{x}\left[\text{ṛ} \xrightarrow{1} \text{ar}\right] + \text{ī} \end{cases}$	tum(x)
vṝ	*vari* *varī*	*varitum* *varītum*
kṝ	*kari* *karī*	*karitum* *karītum*

Case III:

If x ϵ {gup}, then

$$\text{tum(x)} = \begin{cases} \text{x}\left[u \xrightarrow{2} o\right] + i + tum \\ \text{x}\left[u \xrightarrow{2} o\right] + 0 + tum \\ \text{x}\left[u \xrightarrow{2} o\right] + \text{ā}y + i + tum \end{cases}$$

[1] The number of times these words appear in *Aṣṭādhyāyī*; in
ca
A
S
P
P
2(

[2] An exhaustive list of cases for *Tumun* and *san pratyayas* including the multivalued cases are given in the appendix in Devanagari script.

x	$\begin{cases} x\left[u \overset{2}{\rightarrow} o\right] + i \\ x\left[u \overset{2}{\rightarrow} o\right] + 0 \\ x\left[u \overset{2}{\rightarrow} o\right] + \bar{a}y + i \end{cases}$	*tum*(x)
gup	*gopi* *gop* *gopāyi*	*gopitum* *goptum* *gopāyitum*

Case IV:
If x ϵ {tṛp dṛp}, then

$$\text{tum(x)} = \begin{cases} \text{x}\left[\text{ṛ} \overset{2}{\rightarrow} \text{ar}\right] + 0 + \text{tum} \\ \text{x}\left[\text{ṛ} \overset{2}{\rightarrow} \text{r}\right] + 0 + \text{tum} \\ \text{x}\left[\text{ṛ} \overset{2}{\rightarrow} \text{ar}\right] + \text{i} + \text{tum} \end{cases}$$

x	$\begin{cases} x\left[\text{ṛ} \overset{2}{\rightarrow} ar\right] + 0 \\ x\left[\text{ṛ} \overset{2}{\rightarrow} r\right] + 0 \\ x\left[\text{ṛ} \overset{2}{\rightarrow} ar\right] + i \end{cases}$	tum(x)
dṛp	*darp* *drap* *darpi*	*darpatum* *draptum* *darpitum*

Some Multivalued functions for *San Pratyaya*

Case I:
If x'(1)=c, x'(2)=v= *i u*, x'(3)=c in x(which has exactly 3 letters), then

$$\text{san(x)} = \begin{cases} \text{T(x)} + \text{v}'(1) + \text{x}[i\ u \rightarrow e\ o] + iṣa \\ \text{T(x)} + v'(1) + x + iṣa \end{cases}$$

where, $\text{T(x)} = c'(1)[k\ g\ bh\ ṣ\ h\ \rightarrow\ c\ j\ b\ s\ j]$

x	T(x) + v′(1)	*san*(x)
gud	*ju*	*jugodiṣa* *jugudiṣa*
yut	*yu*	*yuyotiṣa* *yuyutiṣa*
vith	*vi*	*vivethiṣa* *vivithiṣa*
cit	*ci*	*cicetiṣa* *cicitiṣa*

Case II:
If there is only one v in x, such that x(2)=v= *i u* and starts with at least two consonants i.e x'(1)=c, x'(2)=c, then

san(x)=

$$\begin{cases} \text{c}'(1) + \text{v}'(1) + x\{v[i\ u \overset{2}{\rightarrow} e\ o\ ]\} + p(x) + ṣa \\ \text{c}'(1) + \text{v}'(1) + x + p(x) + ṣa \end{cases}$$

x	c′(1) + v′(1)	san(x)
cyut	*cu*	*cucyotiṣa* *cucyutiṣa*
kliś	*ci*	*cicleśiṣa* *cikliśiṣa*

7 Conclusion

According to the mathematical definition of a function, it generates a unique output for every input. However, while mathematically modelling *Pratyayas* in Sanskrit we came across several instances where a single input was generating multiple outputs, which have been represented by multivalued functions.
To ensure brevity, *Pāṇini* has used several tools which have been compared with their equivalent tools in our functional approach.

Functions	***Pāṇinian* tools**
x(2)	*upadhā*
c'1,c'2,…,v'1 if x'1=consonant; c'1,c'2,…,v'2 if x'1=vowel	*ekāc*
Multivalued functions	*vā, vibhaṣā, anyatarasyām*
x(1)	*antya*
-	*anuvṛtti*

Table 1: *Pāṇinian* Techniques vs functions

What we are essentially denoting as x(2) in our functions i.e. the penultimate term is nothing but *upadhā*. *Pāṇini* by convention treats x(1) as the end and calls it *antya*. This is clear from the definition of *upadhā* given by *Pāṇini* in Aṣṭādhyāyī *sūtra* '1.1.65 *alontyāt pūrva upadhā'*, which means 'The letter immediately preceding the last letter of a word is called penultimate (*upadhā*) (Creative Commons, 2020)'. As stated before in the paper, the words *vā, vibhaṣā,* and, *anyatarasyam* are used by *Pāṇini* to denote optional forms that we have demoted by multivalued functions.

Another important feature of *Pāṇinian* grammar is *anuvṛtti*, which is a technique of carrying some parts of the previous *sūtras* to the next *sūtras*. Due to *anuvṛtti*, the order in which various elements appear in the *sūtra* itself are very important. However, we do not need to define any such equivalent tool in our modeling as long as

we define some global functions and operators such as p(x) and the '+' operator.

By mathematically modeling *pratyayas*, the reason behind use of these techniques employed by *Pāṇini* to ensure brevity becomes very clear.

Mathematical modelling of *Pāṇinian* grammar in this way helps identify some general patterns, each of which is grouped separately as a case in the functions. These patterns are mainly dependent upon the occurrence of certain specific syllables at certain places. However, we observed that there are some *dhātus* which even after fulfilling the conditions given in the cases, give an output which is different from what is observed in the literature. All such cases needed a separate approach. Hence the for the treatment of such cases input sets for those particular cases have been defined.

The knowledge of *Pāṇinian* rules also helps us reduce the number of individual cases that have been constructed for each function. It helps group certain cases together into a single generalized case. For example: instead of writing three individual functions for *i→e*, *u→o*, and *ṛ→ar*, the knowledge of the rules in *Aṣṭādhyāyī* helps to write a general case of the form *i u ṛ → e o ar*.

Writing such functions for all other *pratyaya* functions may lead us towards a global function for *pratyayas* and for other grammatical tools as well. This technique of mathematical modelling is extremely helpful to understand Sanskrit grammar for people who are non-linguists or do not understand the technicalities of Sanskrit grammar. This mathematical model can also form a base for further processing of the grammatical rules for natural language processing of the language with the help of well-defined input and output sets.

Acknowledgments

Our deepest regards to Prof Swapneel Mahajan from the Department of Mathematics, IIT Bombay whose guidance in terms of inputs and ideas have helped shape the concept of these functions.

References

Agrawal, S. S. (2013). Sanskrit as a Programming Language and Natural Language Processing. *Global Journal of Management and Business Studies. Volume 3*.

Briggs, R. (1985). Knowledge Representation in Sanskrit and artificial Intelligence. *The AI Magazine*, 32-39.

Creative Commons. (2020, 1 4). *Ashtadhyayi*. Retrieved from Paniniya Moolstrot: https://ashtadhyayi.github.io/

Devasthali, G. V. (1967). *Anubandhas of Panini.* Poona: W.H.Golay.

Dhaval Patel, D. (n.d.). *Sanskrit Tool*. Retrieved 12 16, 2019, from Sanskrit World: https://www.sanskritworld.in/sanskrittool/SanskritVerb/tiGanta.html

Dikshita, P. (2014). *kavirasayanmityaparanama kridantkoshah prathamo bhagah.* pratibha prakashan.

Goyal, L. (2011). Comparative analysis of printed Hindi and Punjabi text based on statistical parameters. *Information systems for Indian languages, communications in computer and information science, Volume 139, Part 2.* Berlin, Heidelberg: Springer.

Huet, G. (2003). Lexicon-directed segmentation and tagging in Sanskrit. (pp. 307-325). Helsinki, Finland: In XIIth World Sanskrit Conference.

Kallrath, J. (2013). *Modeling Languages in Mathematical Optimization.* Springer Science & Business Media.

Kulkarni, A. (2016). Brevity in Pāṇini's Aśṭādhyāyī. In B. A. Joseph, *The Interwoven World: Ideas and Encounters in History.* Common Ground Publishing.

S.C.Vasu. (1905). *The Siddhanta Kaumudi of Bhattoji Dikshita.* Allahabad, The Panini office.

Sayanacarya. (1964). *The Madhaviya Dhatuvritti.* Prachya Bharati Prakashan.

Scharf, P. M. (2008). Modeling Paninian Grammar. *International Sanskrit Computational Linguistics Symposium*, (p. 97).

Sharma, D. N. (2018). *Introduction To Panini's Grammar.* CC0 1.0 Universal.

Sohoni, S., & Kulkarni, M. (2018). A Functional Core for the Computational Aṣṭādhyāyī. *Computational Sanskrit and Digital Humanities, Selected papers presented at the 17th World Sanskrit Conference*.

Source, O. (2020). *१.२.३ विभाषोर्णोः*. Retrieved from Ashtadhyayimulstrota: https://ashtadhyayi.github.io//sutra-details/?sutra=1.2.3

Tao Gong, L. S. (2013). Modelling language evolution: Examples and predictions. *Elsivier*, 2.

Tapaswi, N., & Jain, S. (2012). Treebank based deep grammar acquisition and Part-Of-Speech Tagging for Sanskrit sentences. *IEEE*.

Whitney, W. D. (1885). *The roots, verb-forms, and primary derivatives of the Sanskrit language. A supplement to his Sanskrit grammar.* Leipzig, Breitkopf and Härtel.

Williams, M. (2008 revised). *Monier Williams Dictionary.*

Automatic Hadith Segmentation using PPM Compression

Taghreed Tarmom
School of Computing
University of Leeds
Leeds, UK
sctat@leeds.ac.uk

Eric Atwell
School of Computing
University of Leeds
Leeds, UK
e.s.atwell@leeds.ac.uk

Mohammad Alsalka
School of Computing
University of Leeds
Leeds, UK
M.A.Alsalka@leeds.ac.uk

Abstract

In this paper we explore the use of Prediction by partial matching (PPM) compression based to segment Hadith into its two main components (Isnad and Matan). The experiments utilized the PPMD variant of the PPM, showing that PPMD is effective in Hadith segmentation. It was also tested on Hadith corpora of different structures. In the first experiment we used the non-authentic Hadith (NAH) corpus for training models and testing, and in the second experiment we used the NAH corpus for training models and the Leeds University and King Saud University (LK) Hadith corpus for testing PPMD segmenter. PPMD of order 7 achieved an accuracy of 92.76% and 90.10% in the first and second experiments, respectively.

1 Introduction

Automated text segmentation is the task of building a tool that can automatically identify sentence boundaries in a given text and divide them into their components. The need to convert unstructured text into a structured format is especially important when dealing with unstructured text such as web text or old documents.

One of the most important types of old holy Islamic texts in the Arabic language is Hadith. Hadith—the second source of Islam—refers to any action, saying, order, or silent approval of the holy prophet Muhammad that was delivered through a chain of narrators. Each Hadith has an Isnad—the chain of narrators—and a Matan—the act of the prophet Muhammad. Figure 1 shows an example of Hadith.

While most ordinances of Islam are mentioned in the Quran in general terms, detailed and vivid explanations are often provided in the Hadith. This gives the Hadith importance among Muslims. For example, prayer, 'الصلاة', is mentioned in the Quran, while the Hadith specifies what Muslims should do and say; the Hadith explains the time for each prayer and what Muslims should do before and after the prayer. In contrast to the Quran, some Hadiths, which have been handed down over centuries, have been corrupted by incompetent narrators who transferred them incorrectly. Hadith scholars have classified these as non-authentic Hadiths.

Al-Humaydee `Abdullaah ibn Az-Zubayr narrated to us saying: Sufyaan narrated to us, who said: Yahyaa ibn Sa`eed Al-Ansaree narrated to us: Muhammad Ibn Ibraaheem At-Taymee informed me: That he heard `Alqamah Ibn Waqaas Al-Laythee saying: I heard `Umar ibn Al-Khattaab whilst he was upon the pulpit saying: I heard Allaah's Messenger (salallaahu `alaihi wassallam) saying: *"Indeed actions are upon their intentions"*

حَدَّثَنَا الْحُمَيْدِيُّ عَبْدُ اللَّهِ بْنُ الزُّبَيْرِ قَالَ حَدَّثَنَا سُفْيَانُ قَالَ حَدَّثَنَا يَحْيَى بْنُ سَعِيدٍ الأَنْصَارِيُّ قَالَ أَخْبَرَنِي مُحَمَّدُ بْنُ إِبْرَاهِيمَ التَّيْمِيُّ أَنَّهُ سَمِعَ عَلْقَمَةَ بْنَ وَقَّاصٍ اللَّيْثِيَّ يَقُولُ سَمِعْتُ عُمَرَ بْنَ الْخَطَّابِ رَضِيَ اللَّهُ عَنْهُ عَلَى الْمِنْبَرِ، قَالَ سَمِعْتُ رَسُولَ اللَّهِ صَلَّى اللَّهُ عَلَيْهِ وَسَلَّمَ يَقُولُ
"إِنَّمَا الأَعْمَالُ بِالنِّيَّاتِ"

Figure 1: An example of Hadith, Isnad in black and Matan in green.

Automatic Hadith segmentation of Isnad and Matan can help Hadith researchers, some of whom focus on an Isnad with the aim of studying narrators' reliability, the links between them, or how a specific Hadith has been transferred through the ages, sometimes generating a graphical visualization to represent this (Azmi and Badia, 2010). Other research concentrates on Matan to classify Hadiths into topics (Saloot et al., 2016).

Teahan (2000) used prediction by partial matching (PPM) to solve several NLP problems, such as text classification and segmentation. Altamimi and Teahan (2017) and Tarmom et al. (2020b) pointed out that us-

Proceedings of the 17th International Conference on Natural Language Processing, pages 22–29
Patna, India, December 18 - 21, 2020.

ing a character-based compression scheme for tasks such as detecting code-switching and gender/authorship categorization is more effective than word-based machine learning approaches. Many current Hadith studies use a word-based method to segment Hadith from the six canonical Hadith books, but the method of this paper uses a character-based PPM compression method to automatically segment the Isnad and Matan. Our goals are to evaluate PPM segmenter on (1) unstructured Hadith text from lesser-known Hadith books and (2) well-structured Hadith text from the six canonical Hadith books.

This paper explains the data sets chosen for our experiments and outlines the experiments performed on Arabic Hadith text to evaluate the PPM compression method. Finally, we draw conclusions and suggest future work based on this study.

2 Related Work

There have been relatively few studies on the segmentation of Hadith into Isnad and Matan. One study was carried out by Harrag (2014), who developed a finite state transducers-based system to detect the different parts of a Hadith, such as *Title-Bab*, *Num Hadith*, *Sanad* '*Isnad*', and *Matn* '*Matan*'. The disadvantage of this system is that it was built to depend on the Hadith structure in *Sahih Al-Bukhari* book (the most trusted Hadith book), which cannot be used for other Hadith books. Figure 2 shows the Hadith structure in the *Sahih Al-Bukhari* book. This system achieved a precision of 0.44 for Isnad extraction and 0.61 for Matan extraction.

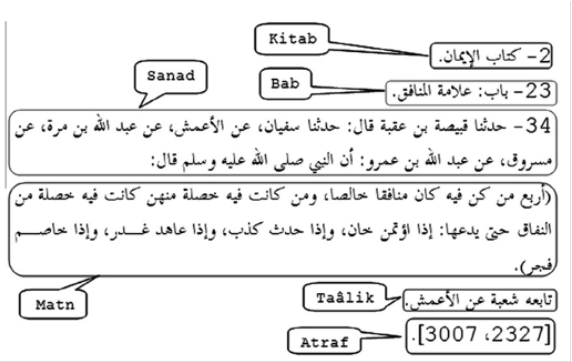


Figure 2: An example of a Hadith structure in the *Sahih Al-Bukhari* book (Harrag, 2014).

Book Name	Precision	Recall	F1 Measure
Sahih Muslim English	96%	91%	93%
Sahih Bukhari English	99%	99%	99%
Sunan Abudawud	100%	100%	100%
Mawta Imam Malik	100%	100%	100%

Table 1: Results of different Hadith books (Mahmood et al., 2018).

Mahmood et al. (2018) selected authentic and reliable Hadith sources such as *Sahih Al-Bukhari*, *Sahih Muslim English*, and *Sunan Abu Dawud*. Since these books differ in format, structure, length, and content, the researchers used different kinds of regular expressions (Regex) for data extraction. However, Hadith patterns extracted by their system lack detail. The results obtained by their system are summarized in Table 1.

Maraoui et al. (2019) implemented a segmentation tool to automatically segment Isnad and Matan from each text from the *Sahih Al-Bukhari* book. First, they analysed the *Sahih Al-Bukhari* corpus and identified the words that distinguish Isnad from Matan. These words were then added to the trigger word dictionary. This tool achieved a precision of 96%.

Altammami et al. (2019) built a Hadith segmenter using N-grams. The *Sahih Al-Bukhari* book was selected as a training set, and the testing set was manually extracted from the six canonical Hadith books. Their result showed that using bi-grams achieved a much higher accuracy (92.5%) than tri-grams (48%).

Most Hadith segmentation research works have used the six famous Hadith books, called The Authentic Six 'الصحاح الستة'. Hence, there is a shortage of research on lesser-known Hadith books, such as Fake Pearls of the Non-Authentic Hadiths 'اللآلئ المصنوعة في الأحاديث الموضوعة'. These books contain a mixture of authentic and non-authentic Hadiths and do not have a clear structure, which makes the segmentation task more complex. Also, character-based text compression methods have not been used in previous Hadith segmentation studies. Our work seeks to fill these gaps in research.

3 Data Collection

For this study, we selected a non-authentic Hadith (NAH) corpus built by Tarmom et al. (2020a) as a training and testing set. The main feature of this corpus is that it contains

452,624 words from different lesser-known Hadith books. It also included several annotated Hadith books, which help to determine the switch points between the Isnad and the Matan, and thus provide a ground truth. Table 2 shows the NAH corpus contents.

These books were downloaded from Hadith websites such as `islamweb.net` and `almeshkat.net` as Word files and converted to csv files. Some of these books have both Hadiths (authentic and NAH), while others only contain NAH. The annotating process was done to determine eight primary features for each Hadith in this corpus. These are *No.*, *Full Hadith*, the *Isnad*, the *Matan*, the *Authors Comments*, the *Hadith Type*, *Authenticity* and *Topic*. A description of the NAH corpus features is shown in Table 3.

4 PPM Compression-based Segmenter

The PPM text compression algorithm is a character-based model that predicts an upcoming symbol by using the previous symbols with a fixed context. Every possible upcoming symbol is assigned a probability based on the frequency of previous occurrences. If a symbol has not been seen before in a particular context, the method will 'escape' to another lower-order context to predict the symbol. This is called the escape method and is used to combine the predictions of all character contexts (Cleary and Witten, 1984). Different variants of PPM have been produced in order to give better compression results, such as PPMC (Moffat, 1990) and PPMD (Howard, 1993). Howard (1993), who invented PPMD, showed that PPMD gives better results for text compression than PPMC.

Equation 1 defines how PPMD estimates the probability P for the next symbol ϕ:

$$P(\phi) = \frac{2C_d(\phi) - 1}{2T_d} \quad (1)$$

where d is the coding order, T_d indicates how many times that the current context, in total, has existed, and $C_d(\phi)$ is the total number of instances for the symbol ϕin the current context. Equation 2 defines how PPMD estimates the escape probability e:

$$e = \frac{t_d}{2T_d} \quad (2)$$

where t_d represents how many times that a unique character has existed following the current context.

Table 4 describes how PPM handles the string 'PXYZXY' with order k=2. For illustration purposes, two has been chosen as the model's maximum order. In order two, if the symbol 'Z' follows the context 'PXYZXY', its probability will be $\frac{1}{2}$ because it has been found before ($XY \rightarrow Z$). The encoding of the symbol 'Z' requires -log($\frac{1}{2}$)=1 bit.

If the symbol 'T' follows the context 'PXYZXY', an escape probability of $\frac{1}{2}$ will be arithmetically encoded because it has not been found after 'XY' in order two. Then the PPM algorithm will move to the lower order, which is order one. In order one, because the symbol 'T' has not been found after the symbol 'Y', an escape probability of $\frac{1}{2}$ will be also encoded. Then, it will be repeated in order zero and an escape probability of $\frac{4}{10}$ will be be encoded because the symbol 'T' has not been found in order zero. Finally, the algorithm will move to order –1. In this order, all symbols are found and the probability will be $\frac{1}{|A|}$, where A = 256 (the alphabet size for ASCII), so its probability will be $\frac{1}{256}$. The encoding of the symbol 'T' requires -log($\frac{1}{2}\times\frac{1}{2}\times\frac{4}{10}\times\frac{1}{256}$)=11.32 bits.

Tawa is a compression-based toolkit that adopts the PPM algorithm. It consists of nine main applications, such as `classify`, `codelength`, `train`, `markup`, and `segment` (Teahan, 2018). This study concentrates on two applications provided by the Tawa toolkit: building models and text segmentation.

4.1 Viterbi Algorithm

For text segmentation using Tawa, we used the toolkit's `train` tool to train multiple models on representative text under research. We then used the markup tool, which utilizes the Viterbi algorithm (Viterbi, 1967). This uses a trellis-based search (Ryan and Nudd, 1993) to find the segmentation with the best compression with all possible segmentation search paths extended at the same time, discarding the poorly performing alternatives (Teahan, 2018).

Figure 3 shows an illustrative example of the search tree for the text segmentation problem in the Tawa toolkit. In this example, the

No.	Book Reference Name	Book's Title	Author	Book's Contents	Hadith's Type	No. of words
1	N1	الأباطيل والمناكير والصحاح والمشاهير	أبو عبد الله الهمذاني الجورقاني	Isnad/Matan/Comments	Authentic and NAH	121,080
2	N2	مائة حديث ضعيف وموضوع منتشرة بين الخطباء والوعاظ	إحسان العتيبي	Matan/Comments	NAH	2,898
3	N3_1	اللآلئ المصنوعة في الأحاديث الموضوعة الجزء الثاني ط دار المعرفة	جلال الدين السيوطي	Isnad/Matan/Comments	Authentic and NAH	15,421
4	N3_2	اللآلئ المصنوعة في الأحاديث الموضوعة الجزء الثاني ط دار المعرفة	جلال الدين السيوطي	Isnad/Matan/Comments	Authentic and NAH	151,382
5	N4	الأحاديث الضعيفة في كتاب رياض الصالحين	إحسان العتيبي	Isnad/Matan/Comments	NAH	5,675
6	N5	الجد الحثيث في بيان ما ليس بحديث ت بو زيد دار الراية	أحمد بن عبد الكريم العامري	Matan/Comments	NAH	16,382
7	N6	الفوائد المجموعة في الأحاديث الموضوعة ط العلمية	الإمام محمد بن علي الشوكاني	Matan/Comments	NAH	139,786

Table 2: The NAH corpus contents.

Features	Description
No.	The Hadith reference number.
Full Hadith	The Hadith as it appears in the book without annotations
Isnad	The chain of narrators
Matan	The act of the Prophet Muhammad
Authors Comments	The author describes the authenticity of each Hadith
Hadith Type	The Hadith Type (Maqtu' مقطوع, Mawquf موقوف and Marfo مرفوع) or Hadith degree (موضوع،ضعيف and so on)
Authenticity	Whether this Hadith is authentic or non-authentic
Topic	The chapter title

Table 3: Features of the NAH corpus.

Order k=2			*Order k=1*			*Order k=0*			Order k=-1		
Prediction	c	p	Prediction	c	p	Prediction	c	p	Prediction	c	p
$PX \rightarrow Y$	1	1/2	$P \rightarrow X$	1	1/2	P	1	1/10	A	1	1/|A|
$\rightarrow Esc$	1	1/2	$\rightarrow Esc$	1	1/2	X	2	2/10			
$XY \rightarrow Z$	1	1/2	$X \rightarrow Y$	2	2/3	Y	2	2/10			
$\rightarrow Esc$	1	1/2	$\rightarrow Esc$	1	1/3	Z	1	1/10			
$YZ \rightarrow X$	1	1/2	$Y \rightarrow Z$	1	1/2	$\rightarrow Esc$	4	4/10			
$\rightarrow Esc$	1	1/2	$\rightarrow Esc$	1	1/2						
$ZX \rightarrow Y$	1	1/2	$Z \rightarrow X$	1	1/2						
$\rightarrow Esc$	1	1/2	$\rightarrow Esc$	1	1/2						

Table 4: Handling the string 'PXYZXY' using PPM with order 2.

tree has a branching of two, since two labels have been used: Isnad and Matan. The label <I> (used for the Isnad model) and <M> (used for the Matan model) show the transformed sequences within each node. If a character switches from one model to the other, the sentinel character is encoded. The compression codelength is also calculated for the transformed sequence, which it appears on the right of each node and below the last nodes. The smallest one, which is the best segmented, is shown in bold font.

5 Evaluation Experiments

Two experiments were performed as part of the evaluation of the compression-based method (provided by Tawa) (Teahan, 2018) to automatically separate Hadith into two components, Isnad and Matan. In the first experiment we used the NAH corpus for training models and testing, and in the second experiment we used the NAH corpus for training models and the Leeds University and King Saud University (LK) Hadith corpus, built by Altammami et al. (2020), for testing PPM segmenter.

5.1 First Experiment

In this experiment, the first book in the NAH corpus, *N1*, was chosen for training purposes. This book is called the *False, Disreputable, and Well-known Hadith Texts* 'الأباطيل والمشاهير والمناكير للجورقاني'. It consists of 732 Hadiths and 121,080 words. Isnads and Matans were manually extracted from *N1* for Isnad and Matan training models, which were 52,221 and 33,489 words long, respectively. The testing text was manually extracted from the third book in NAH corpus, *N3_1*, which contained just Isnad and Matan and is 6,339 words long.

For automatic Hadith segmentation, different orders of PPMD were performed, from order 3 to order 10. As shown in Table 5, Order 7 obtained a higher accuracy of 92.76%, a higher average recall of 0.9365, a higher average precision of 0.9231, and a higher average F-measure of 0.9288. A sample output from the first experiment is shown in Figure 4. Figure 5 shows the last part of Isnad texts were predicted as Matan such as 'عن جابر عن النبي صلى الله عليه وسلم It has been narrated on the authority of Jabir on the authority of the Prophet, may God bless him and grant him peace'(highlighted in blue).

We noticed that the structure of the Isnad texts used in the training set and the testing set differed, creating some confusion in the result. The type of Hadith is given at the beginning of each Hadith in *N1*, for example حديث مرفوع 'Marfo Hadith', which was not labelled as

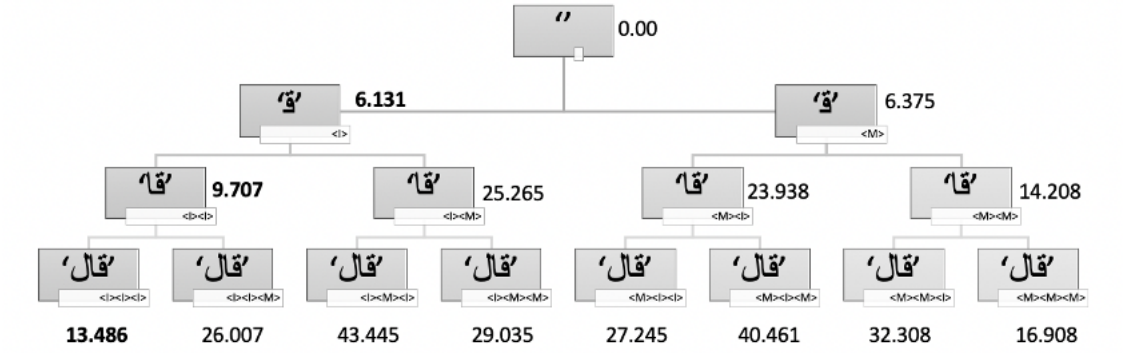


Figure 3: An illustrative example of the search tree for the text segmentation problem in the Tawa toolkit.

Orders	Accuracy (%)	Recall	Precision	F-measure
2	83.34	0.8580	0.8555	0.8568
3	87.20	0.8914	0.8801	0.8858
4	88.04	0.8996	0.8843	0.8919
5	87.02	0.8881	0.8800	0.8840
6	88.58	0.9022	0.8901	0.8961
7	**92.76**	**0.9365**	**0.9231**	**0.9288**
8	92.68	0.9356	0.9222	0.9345
9	92.67	0.9350	0.9215	0.9282
10	91.78	0.9275	0.9127	0.9200

Table 5: Hadith segmentation using PPMD.

<Isnad>حدثنا المقدام بن داود حدثنا أسد بن
موسى حدثنا يوسف بن زياد عن عبدالمنعم بن
إدريس عن أبيه إدريس عن جده وهب بنم منبه عن
أبي هريرة<\Isnad><Matan> أن رجلاً من اليهود
أتى النبي صلى الله عليه وسلم فقال يا رسول الله هل
احتجب الله من خلقه بشيء غير السموات قال نعم
بينه وبين الملائكة الذين حول العرش سبعون
حجاباً من نور وسبعون حجاباً من نار وسبعون
حجاباً من ظلمة وسبعون حجاباً من رفارف الإستبرق
وسبعون حجاباً من رفارف السندس وسبعون حجاباً
من در أبيض وسبعون حجاباً من در أحمر وسبعون
حجاباً من در أصفر وسبعون حجاباً من در أخضر
وسبعون حجاباً من ضياء استضاء من ضوئه النار
والنور وسبعون حجاباً من ثلج وسبعون حجاباً من
ماء وسبعون حجاباً من برد غمام وسبعون حجاباً
من برد وسبعون حجاباً من عظمة الله التي لا توصف
قال فأخبرني عن ملك الله الذي يليه فقال النبي
صلى الله عليه وسلم أصادقت فيما أخبرتك يا يهودي
قال نعم قال فإن الملك الذي يليه إسرافيل ثم
جبريل ثم ميكائيل ثم ملك الموت
<\Matan>

Figure 4: Sample output using PPMD with Order 7.

Isnad (see Figure 6). In the *N3_1* book, each type of Hadith has been written at the end of the Isnad (see Figure 7). Figure 8 shows that Isnad and Matan are correctly predicted but the word مرفوعا 'Marfo' was wrongly predicted

<Isnad>وقال الخطيب أخبرني القيني أنبأنا محمد
بن العباس أنبأنا أبو أيوب سليمان بن إسحق
الحلاب قال سئل إبراهيم الحربي عن حديث موسى بن
إبراهيم عن ابن لهيعة عن أبي الزبير<\Isnad>
<Matan> عن جابر عن النبي صلى الله عليه وسلم من
قال القرآن مخلوق فقد كفر
<\Matan>
<Isnad>حدثنا محمد بن أحمد الوراق حدثنا سعيد
بن محمد ثواب بكر بن عيسى عن محمد بن عثمان
الحراني عن مالك بن دينار عن الحسن ع<\Isnad>
</\Isnad><Matan>ن أنس مرفوعاً أن لله لوحاً أحد
وجهيه درة والآخر ياقوتة قلمه النور فيه يخلق
وبه يرزق وبه يحيي وبه يميت ويعز ويذل ويفعل
ما يشاء في يوم وليلة
<\Matan>

Figure 5: An example of confusion between an Isnad and a Matan using PPMD with Order 7.

as belonging to a Matan since it did not appear in the Isnad training set (highlighted in blue).

حديث مرفوع فقال: فيما أخبر أبو الفضل محمد بن طاهر بن علي المقدسي، رضي الله عنه، قال: أخبرنا علي بن أحمد بن البندار، قال: حدثنا أبو طاهر محمد بن العباس المخلص، قال: حدثنا عبد الله بن محمد بن عبد العزيز البغوي، قال: حدثنا أبو خيثمة زهير بن حرب، قال: حدثنا إسماعيل بن إبراهيم، عن عبد العزيز بن صهيب، عن أنس، عن النبي صلى الله عليه وسلم، قال: "" من كذب علي متعمدا فليتبوأ مقعده من النار "". هذا حديث صحيح، أخرجه الإمام أبو الحسين مسلم بن الحجاج النيسابوري في صحيحه، عن أبي خيثمة زهير بن حرب هكذا

Figure 6: An example of Hadith from *N1* book (Hadith's type is in bold).

ابن عدي) حدثنا أحمد بن محمد بن حرب حدثنا ابن حميد عن جرير عن الأعمش عن أبي صالح عن أبي هريرة **مرفوعاً** القرآن كلام الله لا خالق ولا مخلوق من قال غير ذلك فهو كافر. موضوع: آفته ابن حرب وشيخه أيضاً كذاب وهو محمد بن حميد بن حبان.

Figure 7: An example of Hadith from *N3_1* book (Hadith's type is in bold).

<Isnad>حدثنا محمد بن إسماعيل حدثنا مكي بن إبراهيم حدثنا موسى بن عبيدة عن عامر بن الحكم بن ثوبان عن عبدالله بن عمرو بن العاص عن أبي حازم عن سهل بن سعد<\Isnad><Matan> دون مرفوعاً الله تعالى سبعون ألف حجاب من نور وما تسمع نفس شيئاً من حسن تلك الحجب إلا زهقت نفسها قال أبو الشيخ في العظمة ذكر حجب ربنا تبارك وتعالى فبدأ بهذا الحديث ثم بعده<\Matan>

Figure 8: An example of confusion between an Isnad and a Matan, from the first experiment, because of different Hadith structures in training and testing sets.

We classified some Hadiths as hard Hadiths owing to having a story in the Isnad or between Isnad and Matan which makes the segmentation task more complex. There are two different type of these stories: a narrative story and a chronology story. The narrative story refers to any story related to the narrator such as describing where did he live, his age, who did he meet and so on. The chronology story means telling a sequence of events in order (Sternberg, 1990) such as describing the first event which is the prophet Muhammad and his companions' scene, why did he say a certain Hadith or the person/ group of people who came to ask him and then the following event will be the Matan. We labelled the narrative story as Isnad and the chronology story as Matan. Figure 9 shows an example of the narrative story wrongly predicted as Matan.

5.2 Second Experiment

In this experiment, we used Isnad and Matan training models that were produced from the first experiment. The LK Hadith corpus was chosen for testing purposes. It is a parallel corpus of English-Arabic Hadith, containing 39,038 annotated Hadiths from the six canonical Hadith books.

From the LK corpus, we manually extracted

<Isnad>وقال الطبراني حدثنا علي بن سعيد الرازي حدثنا محمد بن حاتم المؤدب حدثنا القاسم بن مالك المزني حدثنا سفيان بن زياد عن عمه سليم بن زياد <\Isnad><Matan>قال لقيت عكرمة مولى ابن عباس فقال لا تبرح حتى أشهدك على هذا الرجل ابن لمعاذ بن عفراء فقال أخبرني بما أخبرك أبوك عن قول رسول الله صلى الله عليه وسلم فقال حدثني أبي أن رسول الله صلى الله عليه وسلم حدثه أنه رأى رب العالمين عز وجل في حظيرة من القدس في صورة شاب عليه تاج <\Matan>

Figure 9: An example of the narrative story wrongly predicted as Matan using PPMD with Order 7 (highlighted in blue).

chapters two and three from the *Sahih Al-Bukhari* book, comprising a testing file of 10,539 words. We noticed that the last part of Isnads, such as النبي صلى الله عليه وسلم قال 'the Prophet, may God bless him and grant him peace, said', were labelled as Matan so we relabelled these parts as Isnad for consistency with the labelling throughout. Then we removed Arabic diacritics (Al-Tashkeel) and quotation marks.

Order 7 was chosen since it had a higher accuracy rate in the first experiment. The Hadith segmentation using PPMD produced an accuracy of 90.10%, an average precision of 0.9249, an average recall of 0.8607, and an average F-measure of 0.8914. Figure 10 shows the confusion matrix of this experiment and Figure 11 shows an example of the chronology story correctly predicted as Matan from this experiment.

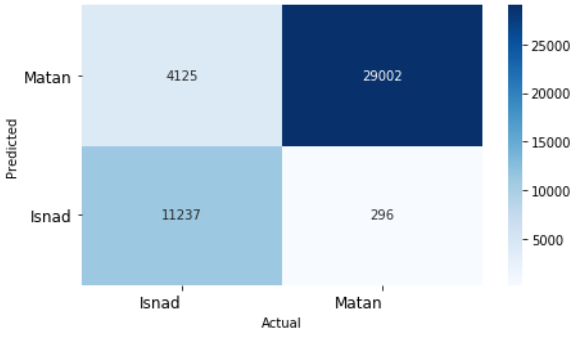


Figure 10: Confusion matrix of the second experiment's results.

6 Conclusion and Future Work

In this paper, we evaluated PPM compression-based method for automatic segmentation of

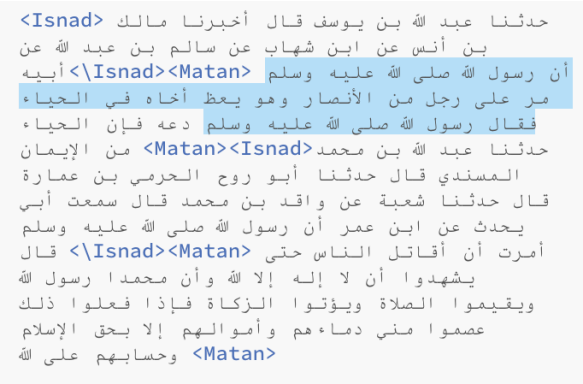


Figure 11: An example of the scene's story correctly predicted as Matan from the second experiment (highlighted in blue).

Arabic Hadith. The experiments showed that PPMD is effective in segmenting Hadith into its two main components (Isnad and Matan), having been tested on Hadith corpora (NAH and LK) that have different structures. The main innovation in these experiments is their use of a character-based text compression method to segment Hadith.

For training Isnad and Matan models we used the first book in the NAH corpus. In the first experiment, we used the third book in the NAH corpus, which lacks a clear structure, as a testing set. We found that PPMD of order 7 obtained a higher accuracy (92.76%) than other orders. In the second experiment, we aimed to evaluate PPMD segmentation on a different Hadith corpus so we used the *Sahih Al-Bukhari* book of the LK Hadith corpus for testing purposes, which produced an accuracy of 90.10%.

The first experiment showed that the Hadith's type is not in the same place between the training and testing set, which leads to some confusion between Isnad and Matan. Possible ways to reduce this confusion that could be undertaken in future work may be to (1) extend the Isnad training set to have different Isnads structured from different Hadith books, (2) clean the testing set from all non-Isnad words.

Acknowledgments

The first author is grateful to the Saudi government for their support.

References

Mohammed Altamimi and William Teahan. 2017. Gender and authorship categorisation of arabic text from twitter using ppm. *International Journal of Computer Science and Information Technology*, 9:131–140.

Shatha Altammami, Eric Atwell, and Ammar Alsalka. 2019. Text segmentation using n-grams to annotate hadith corpus. In *Proceedings of the 3rd Workshop on Arabic Corpus Linguistics*, pages 31–39, Cardiff, United Kingdom. Association for Computational Linguistics.

Shatha Altammami, Eric Atwell, and Ammar Alsalka. 2020. The arabic–english parallel corpus of authentic hadith. *International Journal on Islamic Applications in Computer Science And Technology*, 8(2).

Aqil Azmi and Nawaf Bin Badia. 2010. itree - automating the construction of the narration tree of hadiths (prophetic traditions). In *Proceedings of the 6th International Conference on Natural Language Processing and Knowledge Engineering(NLPKE-2010)*, pages 1–7.

John Cleary and Ian Witten. 1984. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32(4):396–402.

Fouzi Harrag. 2014. Text mining approach for knowledge extraction in sahîh al-bukhari. *Computers in Human Behavior*, 30:558–566.

Paul Glor Howard. 1993. The design and analysis of e cient lossless data compression systems. *Diss. PhD thesis, Brown University.*

Ahsan Mahmood, Hikmat Ullah Khan, Fawaz K Alarfaj, Muhammad Ramzan, and Mahwish Ilyas. 2018. A multilingual datasets repository of the hadith content. *International Journal of Advanced Computer Science and Applications*, 9(2).

Hajer Maraoui, Kais Haddar, and Laurent Romary. 2019. Segmentation tool for hadith corpus to generate tei encoding. In *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2018*, pages 252–260, Cham. Springer International Publishing.

Alistair Moffat. 1990. Implementing the ppm data compression scheme. *IEEE Transactions on Communications*, 38(11):1917–1921.

Matthew S Ryan and Graham R Nudd. 1993. The viterbi algorithm. *University of Warwick. Department of Computer Science.*

Mohammad Arshi Saloot, Norisma Idris, Rohana Mahmud, Salinah Ja'afar, Dirk Thorleuchter, and Abdullah Gani. 2016. Hadith data mining

and classification: a comparative analysis. *Artificial Intelligence Review*, 46(1):113–128.

Meir Sternberg. 1990. Telling in time (i): Chronology and narrative theory. *Poetics Today*, 11(4):901–948.

Taghreed Tarmom, Eric Atwell, and Mohammad Alsalka. 2020a. Non-authentic hadith corpus: Design and methodology. *International Journal on Islamic Applications in Computer Science And Technology*, 8(3).

Taghreed Tarmom, William Teahan, Eric Atwell, and Mohammad Ammar Alsalka. 2020b. Compression versus traditional machine learning classifiers to detect code-switching in varieties and dialects: Arabic as a case study. *Natural Language Engineering*, page 1–14.

William John Teahan. 2000. Text classification and segmentation using minimum cross-entropy. In *Content-Based Multimedia Information Access - Volume 2*, RIAO '00, page 943–961, Paris, FRA. Le Centre De Hautes Etudes Internationales D'informatique Documentaire.

William John Teahan. 2018. A compression-based toolkit for modelling and processing natural language text. *Information*, 9(12):294.

A. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269.

Using multiple ASR hypotheses to boost i18n NLU performance

Charith Peris **Gokmen Oz** **Khadige Abboud** **Venkata sai Varada**

Prashan Wanigasekara **Haidar Khan**

Alexa AI, Amazon
Cambridge MA
{perisc, ogokmen, abboudk, vnk, wprasha, khhaida}@amazon.com

Abstract

Current voice assistants typically use the best hypothesis yielded by their Automatic Speech Recognition (ASR) module as input to their Natural Language Understanding (NLU) module, thereby losing helpful information that might be stored in lower-ranked ASR hypotheses. We explore the change in performance of NLU associated tasks when utilizing five-best ASR hypotheses when compared to status quo for two language datasets, German and Portuguese. To harvest information from the ASR five-best, we leverage extractive summarization and joint extractive-abstractive summarization models for Domain Classification (DC) experiments while using a sequence-to-sequence model with a pointer generator network for Intent Classification (IC) and Named Entity Recognition (NER) multi-task experiments. For the DC full test set, we observe significant improvements of up to 7.2% and 15.5% in micro-averaged F1 scores, for German and Portuguese, respectively. In cases where the best ASR hypothesis was not an exact match to the transcribed utterance (mismatched test set), we see improvements of up to 6.7% and 8.8% micro-averaged F1 scores, for German and Portuguese, respectively. For IC and NER multi-task experiments, when evaluating on the mismatched test set, we see improvements across all domains in German and in 17 out of 19 domains in Portuguese (improvements based on change in SeMER scores). Our results suggest that the use of multiple ASR hypotheses, as opposed to one, can lead to significant performance improvements in the DC task for these non-English datasets. In addition, it could lead to significant improvement in the performance of IC and NER tasks in cases where the ASR model makes mistakes.

1 Introduction

Recent years have seen a dramatic increase in the adoption of intelligent voice assistants such as Amazon Alexa, Apple Siri and Google Assistant. As use cases expand, these assistants are expected to process ever more complex user utterances and perform many different tasks. Some of the key components that enable the performance of these tasks are housed within the spoken language understanding (SLU) system; one being the Automatic Speech Recognition (ASR) module which transcribes the users' vocal sound wave into text and another being the Natural Language Understanding module which performs a variety of downstream tasks that help identify the actions requested by the user (Ram et al., 2018; Gao et al., 2018). These modules perform in tandem and are crucial for the successful processing of user utterances. Typical ASR models generate multiple hypotheses for an input audio signal, that are ranked by their confidence scores (Li et al., 2020). However, only the top ranked hypothesis (referred to hereafter as the ASR 1-best) is usually processed by the NLU module for downstream tasks (Li et al., 2020).

Three major tasks performed by the NLU module are Domain Classification (DC), Intent Classification (IC) and Named Entity Recognition (NER). DC predicts the domain relevant to the utterance (Weather, Shopping, Music etc.) and IC extracts actions requested by users (some examples are, buy an item, play a song or set a reminder). NER is focused on identifying and extracting entities from user requests (names, dates, locations, etc.). Current NLU models usually take in the ASR 1-best hypothesis as input to perform NLU recognition (Li et al., 2020). However, the highest-scored ASR hypothesis is not always correct and, at times, can lead to downstream failures including incorrect NLU hypotheses. These errors can be mitigated by uti-

Proceedings of the 17th International Conference on Natural Language Processing, pages 30–39
Patna, India, December 18 - 21, 2020.

lizing multiple top-ranked ASR hypotheses (ASR n-best hypotheses) in NLU modeling, which have a higher likelihood of containing the correct hypothesis. Even in the case of all n-best hypotheses being incorrect, the NLU models may be capable of recovering the correct hypothesis by integrating the information contained within the n-best hypotheses. Hence, the use of multiple hypotheses should help obtain firmer predictions from ASR modules for their corresponding NLU module and result in improved performance.

In this study we focus on two non-English internal datasets, German and Portuguese, and evaluate the use of ASR n-best hypotheses for improving NLU modeling within these contexts. Given that the ASR models we use in this experiment produce a maximum of five (or less) hypotheses per input utterance, we utilize all available hypotheses (referred to hereafter as the ASR 5-best) for our work. We leverage two BERT-based summarization models (Devlin et al., 2019; Liu, 2019; Liu and Lapata, 2019) and a sequence-to-sequence model with a pointer generator network (Rongali et al., 2020) to extract the information from the ASR 5-best hypotheses. We show that using multiple hypotheses, as opposed to just one, can significantly improve the overall performance of DC, and the performance of IC and NER in cases where the ASR model makes mistakes. We describe relevant work in Section 2 and present a description of our data set and opportunity cost analysis in Section 3. In Section 4 we describe the architecture of our models. In Section 5, we present our experimental results followed by our conclusions in Section 6.

2 Related work

Using deep learning models for summarization has been an active area of research in the recent past. Two popular types in current literature have been extractive summarization and abstractive summarization. Extractive summarization systems summarize by identifying and concatenating the most important sentences in a document whereas abstractive summarization systems conceptualize the task as a sequence-to-sequence problem and generate the summary by paraphrasing sections of the source document. Extensive work has been done on extractive summarization (Liu, 2019; Cheng and Lapata, 2016; Nallapati et al., 2016a; Narayan et al., 2018b; Dong et al., 2018; Zhang et al., 2018; Zhou et al., 2018) and abstractive summarization (Narayan et al., 2018a; See et al., 2017; Rush et al., 2015; Nallapati et al., 2016b) used in isolation. Furthermore, studies have shown improvement in summary quality when extractive and abstractive objectives have been used in combination (Liu and Lapata, 2019; Gehrmann et al., 2018; Li et al., 2018).

Liu (2019) proposed a simple, yet powerful, variant of BERT for extractive summarization in which they modified the input sequence of BERT from its original two sentences to multiple sentences. They used multiple classification tokens ([CLS]) combined with interval segment embeddings to distinguish multiple sentences within a document. They appended several summarization specific layers (either a simple classifier, a transformer or an LSTM) on top of the BERT outputs to capture document level features relevant for extracting summaries. Following this work, Liu and Lapata (2019) proposed a model that comprises of the pre-trained BERT extractive summarization model (Liu, 2019) as the encoder and a decoder which consists of a 6-layered transformer (Vaswani et al., 2017). The encoder was fine-tuned in two stages, first on the extractive summarization task and then again on an abstractive summarization task resulting in a joint extractive-abstractive model that showed improved performance on summarization tasks.

The utilization of multiple ASR hypotheses for improved NLU model performance across DC, IC tasks was first introduced by Li et al. (2020). They proposed the use of 5-best ASR hypotheses to train a BiLSTM language model, instead of using a single 1-best hypothesis selected using either majority vote, highest confidence score or a reranker. They explored two methods to integrate the n-best hypothesis: a basic concatenation of hypotheses text and a hypothesis embedding concatenation using max/avg pooling. The results show 14%-25% relative gains in both DC and IC accuracy.

In our work, we explore the performance improvement offered by utilizing the ASR 5-best hypotheses in previously unexplored languages, German and Portuguese. We also differ from previous studies due to our use of the superior BERT-based extractive (Liu, 2019) and joint extractive-abstractive (Liu and Lapata, 2019) summarization models to extract a summary hypothesis for the DC task, from the ASR 5-best.

Voice assistants traditionally handle IC and NER

tasks using semantic parsing components which typically comprise of statistical slot-filling systems for simple queries and, in more recent time, shift-reduce parsers (Gupta et al., 2018; Einolghozati et al., 2019) for more complex utterances. Rongali et al. (2020) proposed a unified architecture based on sequence-to-sequence models and pointer generator networks to handle both simple and complex IC and NER tasks with which they achieve state-of-the-art results. In this work, we use a model that expands this approach to consume the 5-best ASR hypotheses and evaluate its performance on IC/NER tasks for the two language datasets considered.

3 Data

Our experiments focus on two non-English internal datasets; German and Portuguese. We run all utterances in each language through one language-specific ASR model and take the top-ranked ASR hypothesis for each utterance as ASR 1-best and all available hypotheses for each utterance (a maximum of five in our models) as ASR 5-best. In addition, we also obtain a human transcribed version of each utterance. For German, we use 1.48 million utterances from 21 domains for training and validation. We split the data randomly within each domain, with 85% used for training and 15% for validation. An independent set of 193K utterances are used for testing. Within the independent test set we find 17K utterances where the ASR 1-best did not match the transcribed utterance exactly and mark them as the "mismatched" test set. (Table 1). For Portuguese, we use 890K utterances from 19 domains for training and validation, split the same way as with German. Another 247K utterances are used for testing. We find 41K utterances within test, where the ASR 1-best did not match the transcribed utterance exactly, and mark them as the mismatched test set (Table 1).

3.1 Opportunity Cost Measurement

Li et al. (2020) showed improvement in NLU model performance on English (en-US) upon utilizing the ASR 5-best hypotheses instead of only ASR 1-best. However, the impact of this on non-English languages has not yet been explored. To understand the opportunity of improvement that the ASR 5-best hypotheses can lend to NLU model performance in German and Portuguese datasets, we analyze the ASR 5-best hypotheses in comparison to the ground-truth human transcribed data for each of the considered language datasets. First, we calculate the number of exact matches to the transcribed utterance occurring in each of the top 5-best hypotheses. It should be mentioned that each ASR hypothesis is different from the others and only one hypothesis (if at all) can match the transcribed utterance. Next we compute the amount of exact matches found in the n^{th}-best hypothesis set, as a fraction of the volume of exact matches found at 1-best. The results are shown in Table 2. We find that the amount of exact matches that occur in 2-5 best hypotheses, compared to the volume of exact matches that occur in the top-ranked hypothesis, is large for Portuguese (30.16%) and German (20.83%) (see Table 2). This gives an indication of the opportunity present in using hypotheses beyond ASR 1-best for each language dataset.

In Table 3, we further illustrate the use of the ASR 5-best hypotheses by showing three possible cases of stored information that we want our NLU model to extract; selecting the best matching hypothesis (first and second rows) and combining hypotheses (third row).

4 Experimental Setup

4.1 DC models

For our DC experiments, we compare performance across the following classification models:

- **Baseline** – A BERT-based classification baseline model with MLP classifier trained on the *transcribed utterance* and tested on the *ASR 1-best*
- **BSUMEXT**– A BERT-based extractive summarization model trained and tested on the *ASR 5-best*
- **BSUMEXTABS**– A BERT-based joint extractive and abstractive summarization model trained and tested on the *ASR 5-best*

Standard testing on transcribed utterances underestimates the combined ASR and NLU errors. In order to avoid this our test sets exclude transcribed utterances and thus reflect the real situation.

In Section 3, we described the simple extractive summarization model proposed by Liu (2019). We adapt their extractive summarization model to take the ASR 5-best hypotheses as input and output a probability score per domain based on a summarized hypothesis. Figure 1 shows the architecture

Table 1: Total data set sizes in terms of utterance counts

Language	Train	Validation	Test (full)	Test (mismatched)
German	1,255,402	221,543	192,697	16,672
Portuguese	756,148	133,438	246,638	40,896

Table 2: Exact Matches to the transcribed utterance found in ASR n-best as a percentage of Exact Matches found in ASR 1-best

n	**Portuguese (%)**	**German (%)**
2	16.55	10.26
3	7.1	5.01
4	3.92	3.33
5	2.59	2.23
total	30.16	20.83

of BSUMEXT with ASR 5-best input. The task of the BSUMEXT model is to create an extractive summary by picking from the class assigned to each hypothesis. This summary is then fed into a multi-layer perceptron classifier to perform the DC task. As in the case of Liu (2019), vanilla BERT is modified to include multiple [CLS] symbols. Each symbol is used to obtained features of each of the ASR n-best hypotheses preceding it. Alternating hypotheses fed into the model are assigned a segment embedding (E_A or E_B), based on whether it is an even or odd numbered hypothesis. For example for a sentence "play music" :

```
ASR 1-best: play muse      [E_A]
ASR 2-best: play mu chick  [E_B]
ASR 3-best: play news      [E_A]
ASR 4-best: play mus       [E_B]
ASR 5-best: play my sick   [E_A]
```

The model then takes the [CLS] representation of each ASR 5-best utterance and performs multi-headed attention to obtain the summary hypothesis.

For the BSUMEXTABS model, the BERT encoder is fine-tuned on an abstractive summarization task and then further fine-tuned on the extractive summarization task. In this model the summary hypothesis fed into the multi-layer perceptron classifier, is generated token by token in a sequence-to-sequence fashion. Similar to Liu and Lapata (2019), a decoupled fine-tuning schedule which separates the optimizers of the encoder and the decoder is used.

We trained each of our models for up to 30 epochs and use the best performing model, based on validation metrics, for evaluating the independent test set.

4.2 IC/NER models

We compare the following models for the IC and NER tasks:

- **Baseline** – A BERT-based classification baseline model trained on the *transcribed utterance* and tested on the *ASR 1-best*
- **BERT_S2S_NBEST_PTR** – A BERT-based sequence-to-sequence model which employs a pointer generator network, trained on the *ASR 5-best + transcribed utterance* and tested on *ASR 5-best*

Instead of a typical sequence tagging problem, Rongali et al. (2020) propose a unified architecture to handle IC and NER tasks as a sequence generation problem. We build upon that approach. BERT_S2S_NBEST_PTR is a sequence-to-sequence model augmented with a pointer generator network which functions as a self-attention mechanism. We expand the architecture proposed by Rongali et al. (2020) to include multiple input queries. The model task is to generate target words which can be either intent or slot delimiters or words that are from the source sequences. The pointer generator network enables the model to generate pointers to the source sequences (instead of using a large vocabulary of tokens) within the target sequence. An example of a source sequence with two ASR hypotheses and a target sequence looks as follows (we use spaces to delimit hypotheses and _&_ to delimit separate tokens within an utterance):

```
Source: ply_&_madonna play_&_mad_&_owner
Target: PlaySongIntent( @ptr1_0
    ArtistName( @ptr0_1 )ArtistName )
    PlaySongIntent
```

where @ptr0_1, for example, is a pointer to the second word "madonna" in the first utterance of the source query. One advantage of using pointers instead of the actual tokens is the smaller target vocabulary required for the decoder, resulting in a more light-weight model.

The architecture consists of a pre-trained BERT encoder and a transformer decoder (Devlin et al.,

Table 3: Illustrative examples in English that compares the 3-best ASR hypotheses to the transcribed utterance

Transcription	1- best hypothesis	2-best hypothesis	3-best hypothesis
buy movie mystery	**buy movie mystery**	**buy** my tree	but move my tree
who is nelson	how is my son	**who is nelson**	how samsung
play music	pull **music**	pull news	**play** my muse

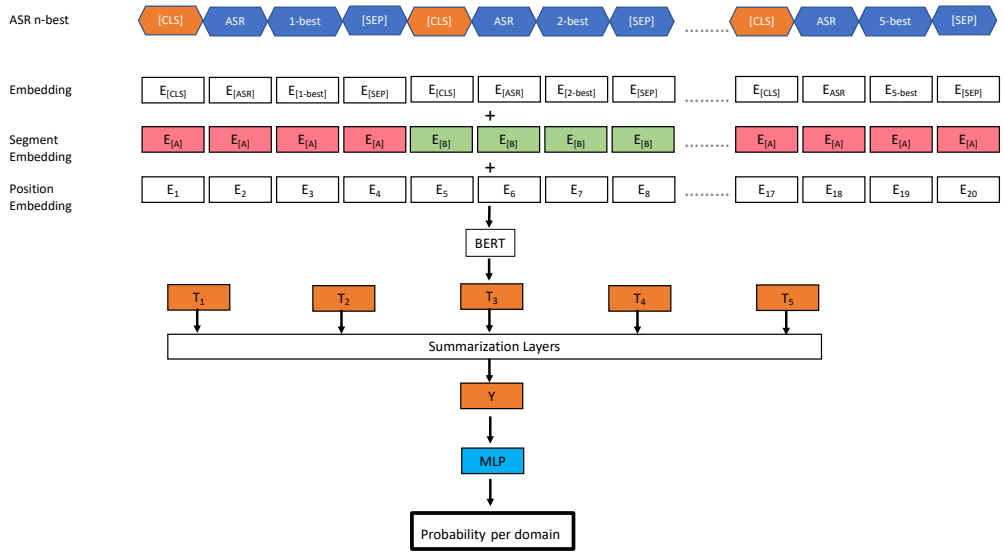


Figure 1: A schematic of the architecture of the BSUMEXT

2019; Vaswani et al., 2017). The decoder is augmented with a pointer generator network that functions as a self-attention mechanism. Figure 2 shows the high-level architecture. The Bert encoder processes each ASR hypothesis separately. The encoder hidden states over all ASR hypotheses are then concatenated and passed to the decoder. The decoder hidden states are used to update the attention mechanism and the tagging vocabulary and pointer distributions (see Rongali et al. (2020) for detailed descriptions). These probability distributions of tags and pointers are used to determine the next word and tag that is output by the decoder. The model is trained by minimizing sequence cross entropy loss over the training set.

These models are domain-specific multi-task models which handle both IC and NER tasks simultaneously. We trained one model per domain with all models trained for up to 50 epochs. The best performing model based on validation metrics was used for evaluating the independent test set.

5 Results and Discussion

5.1 Evaluation

We measure the success of our DC experiments by comparing both micro- and macro-averaged F1 scores of our experimental models to those of the baseline model. Micro- and macro-averaged F1 scores are defined as

$$F1_{micro} = \frac{2 \times P \times R}{P+R} \tag{1}$$

$$F1_{macro} = \frac{1}{n}\sum_i F1_i = \frac{1}{n}\sum_i \frac{2 \times P_i \times R_i}{P_i + R_i} \tag{2}$$

where P and R are overall precision and recall respectively and P_i and R_i are the within class precisions and recalls respectively. We also calculate the relative change in error of each experimental model run with respect to baseline as shown in equation 3. Note that "lower-is-better" for this metric. In addition to these metrics calculated on the *full* test data set, we also calculate these metrics on the *mismatched* test set utterances where the ASR 1-best did not match the transcribed utterance.

$$\Delta_{err} = 100 \times \frac{((100 - F1_{experiment}) - (100 - F1_{baseline}))}{(100 - F1_{baseline}))} \tag{3}$$

For the IC and NER experiments, we use Semantic Error Rate (SemER) (Su et al., 2018) as our metric of choice. SemER is defined as follows:

$$SemER = \frac{D+I+S}{C+D+S} \tag{4}$$

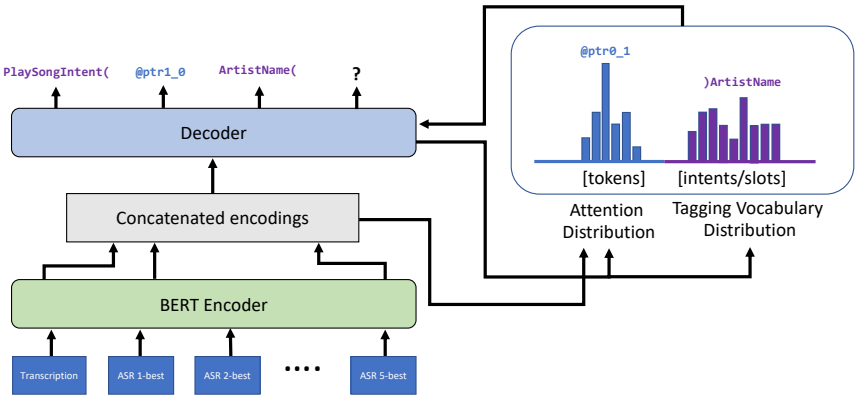


Figure 2: A schematic of the sequence-to-sequence model with attention. Each ASR hypothesis is encoded separately. The encoder hidden states are then concatenated and passed to the decoder to have a cross-attention between encoder and decoder outputs over all ASR hypotheses.

Table 4: Evaluation on the *full* and *mismatched* test sets for DC. Relative change in error rate (Δ_{err}) measured against baseline for each metric is shown in each succeeding column (negative is good).

Model	**Full set**		**Mismatched set**	
	f1_micro (Δ_{err})	**f1_macro** (Δ_{err})	**f1_micro** (Δ_{err})	**f1_macro** (Δ_{err})
German				
BSUMEXT	-1.60%	-4%	-5.40%	-12%
BSUMEXTABS	-7.20%	-3.90%	-6.70%	-2.30%
Portuguese				
BSUMEXT	-12.60%	4.90%	-6.30%	-0.30%
BSUMEXTABS	-15.50%	-7.30%	-8.80%	-7.40%

where D=deletion, I=insertion, S=substitution and C=correct-slots. The Intent is treated as a slot in this metric and Intent error, considered as a substitution. We use the relative change in SemER with respect to the baseline model (equation 5), both overall and per domain in order to evaluate the success of our models. Note that "lower-is-better" for relative change in SemER as well.

$$\Delta_{sem} = 100 \times \frac{(SemER_{experiment} - SemER_{baseline})}{SemER_{baseline}} \quad (5)$$

5.2 DC experiments

Table 4 describes the performance of all the models defined in Section 4.1 on the full test set and the mismatched test set (see Section 3 and Table 1). The full test set enables us to understand the general performance improvement that can be achieved by using summarization models. Although utilizing the full ASR 5-best hypotheses might offer some improvement even in cases where the ASR 1-best hypothesis is an exact match to the transcribed utterance, much more value-add is expected when using the ASR 5-best hypotheses in cases where there is a mismatch between the transcribed utterance and ASR 1-best. To study this use case, we use the mismatched test set.

We observed that a majority of F1 scores across all models for German exceeded their corresponding values in Portuguese. Our opportunity cost analysis showed that exact matches between the transcribed utterance and ASR 2-5-best for Portuguese are higher than for German (see Section 3.1). This suggests that the German ASR model tends to perform better than the Portuguese ASR model. In this light, the smaller gains in relative change in error observed for German when compared to Portuguese are likely due to the German ASR model being superior and therefore leaving smaller room for improvement.

Figure 3 displays the relative changes of each model against the baseline for each dataset. When considering micro-averaged F1 scores, the BSUMEXT and BSUMEXTABS models outperform the baseline in all cases, with the later out-performing the former. This shows that the use of ASR 5-best hypotheses can significantly improve overall classification for both language datasets. The BSUMEXTABS models also consistently out-perform the baseline on macro-averaged F1 scores, showing improvement in mean within-class classification scores as well. This suggests that BSUMEXTABS with additional fine-tuning

on the abstractive task, is in general more successful at creating a firmer hypothesis for DC than the pure extractive summarization of BSUMEXT. For Portuguese, even with the relatively large percentage of exact matches available for extraction within its ASR 2-5 hypotheses (see Section 3), BSUMEXTABS consistently outperforms BSUMEXT across all metrics and datasets.

5.3 IC and NER experiments

Table 5 describes the performance of all the models defined in Section 4.2 on domain-level data from the full test set and the mismatched test set. As with the DC experiments, we use the full test set to understand the general overall performance improvement, and use the mismatched test set to identify improvement in cases where the ASR 1-best hypothesis is not an exact match to the transcribed utterance.

When evaluating the BERT_S2S_NBEST_PTR model, we find that it tends improve performance specifically on the mismatched test set. For German, we find improved performance across every domain on the mismatched test set (see Table 5) with an overall SemER improvement of 11.6% against baseline. However, we only observe improvement in three domains on the full set, while other domains show degradation in SemER. It is also interesting to note that the domains that improve also had low utterance counts. For Portuguese, testing on the mismatched test set yields improved performance across 17 out of 19 domains (see Table 5) with an overall SemER improvement of 8.1% against baseline, while we see only three domains show improvement on the full test set. Our results suggest that the ASR 1-best hypothesis works well for IC/NER tasks. The noise added by additional hypotheses seem to degrade results in the general use case. However, the additional hypotheses tend to be very helpful in cases where the ASR model makes mistakes (i.e. mismatched set data where the ASR 1-best is not an exact match to the transcribed utterance).

Our full test set results show that the baseline model appears to be a better choice for the IC/NER tasks. However, if we could detect user utterances where the ASR model might have made a mistake in its top hypothesis, the ASR outputs (i.e. the set of all hypotheses) of these utterances could be channeled to a separate NLU model such as BERT_S2S_NBEST_PTR, that could build a better

Table 5: Joint evaluation on *full* and *mismatched* test sets for IC/NER tasks. Δ_{sem} (%) is the relative change in SemER against baseline for each domain (negative is good).

German		
Domain	**S2S_NBEST_PTR**	
	Full Set Δ_{sem} (%)	**Mismatched Set Δ_{sem} (%)**
domain A	14.79	-14.16
domain B	30.33	-10.27
domain C	25.25	-5.83
domain D	95.41	-7.3
domain E	16.38	-12.68
domain F	12.54	-18.9
domain G	-33.51	-23.2
domain H	7.41	-14.2
domain I	12.96	-25.2
domain J	15.27	-3.72
domain K	32.02	-7.42
domain L	89.45	-18.63
domain M	643.85	-15.95
domain N	1.06	-7.21
domain O	-34.8	-25.02
domain P	26.52	-8.74
domain Q	8.47	-6.13
domain R	69.35	-13.76
domain S	19.07	-2.12
domain T	-4.25	-10.93
domain U	1.92	-7.33
Overall	19.17	-11.64
Portuguese		
Domain	**S2S_NBEST_PTR**	
	Full Set Δ_{sem} (%)	**Mismatched Set Δ_{sem} (%)**
domain A	2.89	-14.11
domain B	18.88	-7.94
domain C	46.86	-14.7
domain D	4.3	3.16
domain E	-12.54	-30.65
domain F	5.87	-18.89
domain G	6.56	-3.7
domain H	24.64	-2.57
domain I	71.12	-24.42
domain J	-7.69	-10.03
domain K	19.16	-5.45
domain L	11.15	-9.97
domain M	3.54	-10.58
domain N	48.85	-10.15
domain O	-30.38	-59.98
domain P	6.84	-12.29
domain Q	0.11	-15.66
domain R	20.49	-8.94
domain V	1533.33	47.62
Overall	106.58	-8.09

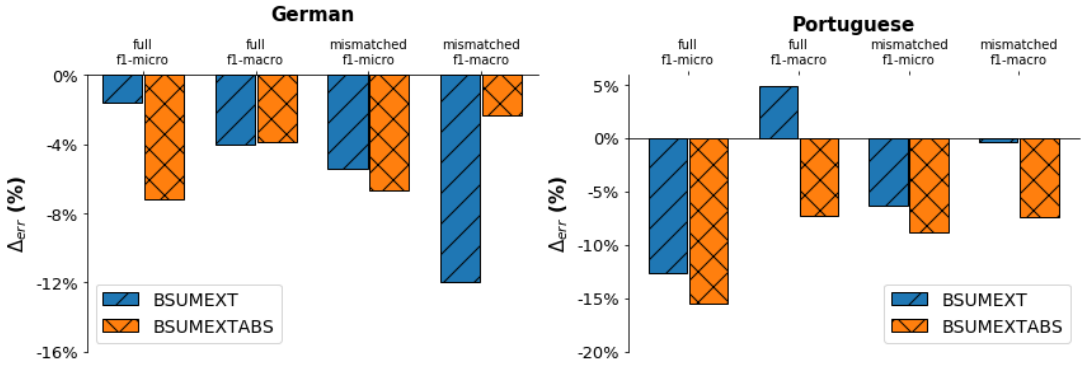


Figure 3: Relative change in error rate measured against baseline for each metric on full and unmatched test sets for DC experiments.

hypothesis than the baseline and improve overall IC/NER performance.

We analyzed the confidence scores of our ASR models on the full and mismatched test set hypotheses to explore the possibility of detecting a mismatched set ASR output. For each ASR output we obtain the mean confidence score across all available hypotheses. We then compare the frequency distributions of the mean confidence scores in the full and mismatched test sets. Figure 4 shows the resulting distributions for two example domains for each language dataset. We find that the full set shows a strong peak at high confidence scores while the mismatched set shows a more uniform distribution. The pronounced difference in distribution shape suggests that a thresholding mechanism based on the confidence score output by the ASR model (or a simple classifier trained on ASR outputs and scores) might be used to predict mismatched test set outputs with good confidence. Leveraging such a mechanism might enable the use of a second model such as BERT_S2S_NBEST_PTR to improve performance in these mismatched cases, and in turn improve overall IC/NER performance.

6 Conclusions and future work

In this study, we explore the benefits of using ASR 5-best hypotheses for the NLU tasks in the German and Portuguese datasets. We explore several models to perform DC and IC/NER tasks and evaluate their performance against baseline models that use ASR 1-best. We find significant overall improvement in performance for the DC task. We also find significant improvement in performance of the jointly evaluated IC/NER tasks in cases where the ASR 1-best hypothesis is not an exact match to the transcribed utterance. For the DC task, our results suggest that the use of ASR 5-best helps produce better hypotheses and thereby greater improvements in the case of slight lower quality ASR models.

Our next steps will include exploring how different data splits based on ASR confidence scores might affect the sequence-to-sequence model performance. Furthermore, we will explore performance improvements in IC and NER tasks, using different model architectures and training schedules. We will also expand our study to a larger set of languages in order to understand how the use of multiple ASR hypotheses might affect languages with different lexical distributions. Languages which use multiple scripts (Japanese, Hindi, Arabic etc.) or which are more opaque and likely to have heterographs (e.g., “serial, “cereal”) and those that have less standardized spelling systems (Hindi etc) are more likely to have ASR errors. They may have different levels of improvement with the use of ASR 5-best hypotheses and we hope to analyze this in our future work.

References

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. *CoRR*, abs/1603.07252.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

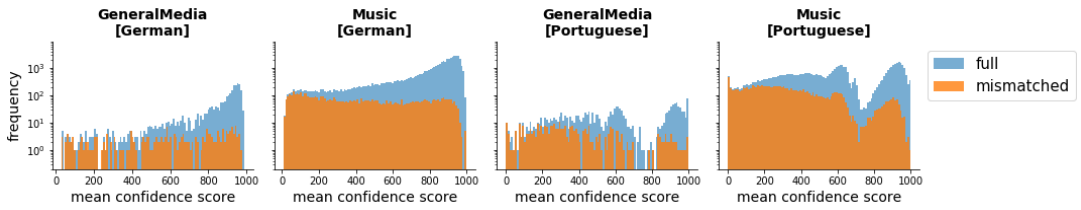


Figure 4: Frequency distributions of mean confidence score across all available hypotheses for each data point in *full* and *mismatched* test sets. We show results for only two domains for each language due to space limitations. The distributions show similar shape across all domains within each language dataset.

Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. 2018. BanditSum: Extractive summarization as a contextual bandit. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3739–3748, Brussels, Belgium. Association for Computational Linguistics.

Arash Einolghozati, Panupong Pasupat, Sonal Gupta, Rushin Shah, Mrinal Mohit, Mike Lewis, and Luke Zettlemoyer. 2019. Improving semantic parsing for task oriented dialog. *CoRR*, abs/1902.06000.

Ge Gao, Eunsol Choi, Yejin Choi, and Luke Zettlemoyer. 2018. Neural metaphor detection in context. *CoRR*, abs/1808.09653.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium. Association for Computational Linguistics.

S. Gupta, Rushin Shah, Mrinal Mohit, A. Kumar, and M. Lewis. 2018. Semantic parsing for task oriented dialog using hierarchical representations. In *EMNLP*.

Mingda Li, Weitong Ruan, Xinyue Liu, Luca Soldaini, W. Hamza, and Chengwei Su. 2020. Improving spoken language understanding by exploiting asr n-best hypotheses. *ArXiv*, abs/2001.05284.

Wei Li, Xinyan Xiao, Yajuan Lyu, and Yuanzhuo Wang. 2018. Improving neural abstractive document summarization with explicit information selection modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1787–1796, Brussels, Belgium. Association for Computational Linguistics.

Yang Liu. 2019. Fine-tune BERT for extractive summarization. *CoRR*, abs/1903.10318.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. *CoRR*, abs/1908.08345.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2016a. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. *CoRR*, abs/1611.04230.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar GuÌ‡lçehre, and Bing Xiang. 2016b. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018a. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018b. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1747–1759, New Orleans, Louisiana. Association for Computational Linguistics.

Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, Eric King, Kate Bland, Amanda Wartick, Yi Pan, Han Song, Sk Jayadevan, Gene Hwang, and Art Pettigrue. 2018. Conversational AI: the science behind the alexa prize. *CoRR*, abs/1801.03604.

Subendhu Rongali, Luca Soldaini, Emilio Monti, and Wael Hamza. 2020. Don't parse, generate! a sequence to sequence architecture for task-oriented semantic parsing. *Proceedings of The Web Conference 2020*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *CoRR*, abs/1509.00685.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. *CoRR*, abs/1704.04368.

Chengwei Su, Rahul Gupta, Shankar Ananthakrishnan, and Spyros Matsoukas. 2018. A re-ranker scheme for integrating large scale NLU models. *CoRR*, abs/1809.09605.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Xingxing Zhang, Mirella Lapata, Furu Wei, and Ming Zhou. 2018. Neural latent extractive document summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 779–784, Brussels, Belgium. Association for Computational Linguistics.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–663, Melbourne, Australia. Association for Computational Linguistics.

A Grammatical Sketch of Asur: A North Munda language

Zoya Khalid
Central University of Jharkhand

Abstract

Asur belongs to North Munda sub-branch of Austro-Asiatic languages which now has less than 10,000 speakers. This is a very first attempt at describing and documenting Asur language, therefore the approach of this paper is descriptive rather than that of answering research questions. The paper attempts to describe the grammatical features such as number, case, pronouns, tense-aspect-mood, negation, question formation, etc. of Asur language. It briefly touches upon the morphosyntactic and typological features of Asur, with the intent to present a concise overview of the language, which has so far remained almost untouched by documentary linguistics.

1 Introduction

The population of Asur (ISO-639-3) speakers in 2007 was 7,000 (Ethnologue, 2018). UNESO (Atlas of the World's Languages in Danger, 2010) has classified Asur language as a 'definitely endangered' one. Asur is also known as Asuri and Ashree. Asur, Ho and Mundari are mutually intelligible as they belong from the group of Kherwarian languages.

2 Data collection and Methodology

Data for the present work has been recorded from native Asur speakers living in Bishunpur block, Gumla district in Jharkhand in India. Data was recorded mostly through interview method. Audio recorders were used for data recording; the audio files were segmented, transcribed and translated using (SayMore) software, which was also used for handling metadata related to Asur fieldwork and data collection, for analysis of data (Fieldworks) software was used (commonly known as FLEx). For transcription IPA (International Phonetic Alphabet); and interlinear glossing was followed for annotation, along with free translation in English.

3 Asur Morphosyntax

3.1 Word Order

Asur is a SOV (Subject Object Verb) language, It is interesting to note that Munda languages have departed from their non-verb final basic word order that is found in majority Austro-Asiatic languages (Jenny, Weber, & Waymuth, 2015) and have shifted to verb final basic word order like Indo-Aryan languages (Subbārāo, 2012). Example (1) is an intransitive sentence in Asur which shows the positioning of subject (S), object (O) and verb (V) in Asur.

1.	iŋ	kadri	ke	lel-l=iŋ
	1S	Kadri	ACC	see-PST=1S
	S	**O**		**V**

"I saw Kandri."

In Asur the Indirect Object (IO) comes before Direct Object (DO), but their positions are interchangeable and shown in (2) and (3).

2.	iŋ	lamta	ke	kitab	ema-l=iŋ
	1S	Lamta	ACC	book	give-PST=1S
	S	**IO**		**DO**	**V**

"I gave book to Lamta."

3.	iŋ	kitab	lamta	ke	ema-l=iŋ
	1S	book	Lamta	ACC	give-PST=1S
	S	**DO**	**IO**		**V**

"I gave book to Lamta."

3.2 Pronouns, demonstratives and pronominal clitics

Pronouns in Asur have three persons and three numbers (Singular, dual and plural). In Asur

Proceedings of the 17th International Conference on Natural Language Processing, pages 40–49
Patna, India, December 18 - 21, 2020.

language 1st person pronouns for Dual and Plural have two further categories- inclusive (including the addressee) and exclusive (excluding the addressee). The 2nd person singular has different forms for non-honorific and honorific category, i.e. /am/ and /ape/ respectively. Table (1) shows the pronouns in Asur language.

Table 1: Pronouns in Asur

Persons	Singular	Dual	Plural
1st	iŋ	alaŋ(inclusive)	abu(inclusive)
		aliŋ(exclusive)	ale(exclusive)
2nd	am	aben	ape
2nd Hon	ape	aben	ape
3rd	ae	akin	aku

Demonstratives are words in a language that are used to indicate a referent's spatial or temporal distance from the spatial or temporal position of the speaker. Table (2) shows the demonstratives in Asur.

Table 2: Asur demonstratives

	Animacy	Singular	Dual	Plural
Prox	A	nui	nokin/nukin	nuku
	IA	neʔa, nea, niya		
Dist	A	huni	hokin/hukin	huku
	IA	hona		

Asur pronominal clitics attach to verbs or other constituents; it bears information about the subject and/or object of the sentence.

3.3 Number

Asur language distinguishes between animate and inanimate nouns. Asur animate nouns take the suffix /=ku/ or /=aku/ to pluralize as in sentence (8). For inanimate nouns quantifiers are used to show the sense of plurality. In Asur plural marker is optionally attached to the noun, it may also manifests on the verb as agreement marker or pronominal marker, without appearing on the noun.

3.4 Quantifiers

Quantifiers are used in a language to indicate quantity. Some of the quantifiers used in Asur are as follows:

1. /ɖʰer/ and /bagra/ "many" or "much"
2. /jama/ "all"
3. /miyaɖmin/ "even a single" or "even one" (generally used in negative sentences.)
4. /katiʔken/ "some/little" or "few" (may be used with count or mass nouns)

3.5 Case and postposition

Asur, being a SOV (Subject Object Verb) language is postpositional language like Indo-Aryan languages. The following are case and postpositional markers in Asur.

3.5.1 Nominative

Nominative case marks the Subject of the verb. Asur does not overtly mark nominative case, as can be seen in (1).

3.5.2 Accusative and Dative

Accusative and dative cases are cases given to noun by the verb. Asur uses /ke/ to mark accusative as well as dative as shown in examples (2) and (4).

4. gʰoɽa **ke** cara eme-me
   horse **DAT** fodder give-IMP
   "Give fodder to the horse."

3.5.3 Locative

Locative case is used to indicate location. Asur locative markers appear as /re/ or /ɽe/.

5. kitab ʈebal **re** edana
   book table **LOC** COP.PRS
   "Book is on the table."

3.5.4 Instrumental

Instrumental case is used most often to indicate use of an instrument for completion of action; in the following example, instrumental case is used in Asur to indicate state of being hungry due to hunger.

6. ram raŋek **te** edaniya
   ram hunger **INST** COP.PRS
   "Ram is hungry"

3.5.5 Comitative

Comitative is an indicator of accompaniment. Sentence (7) exemplifies the use of comitative marker /lo/ in Asur.

7. ceŋa bilai **lo** eneʔ tan=ae
   child cat **COM** play Prog=3S
   "The child is playing with the cat."

3.5.6 Ablative

Asur language uses ablative case markers /hare/ or /hareʔ/ and /tara/ for instances where separation of two people or things is implied, the markers can be used interchangeably.

8. ceŋa=ku oɽaʔ **tara** bahir uɽuŋ-n=aku
   child=3P house **ABL** outside come-ITR=3P
   "Children came out from the house."

3.5.7 Genitive

Genitive markers have three forms in Asur i.e.- /-ala/, /-ali/ and /-ra/ or /-rena/. The marker /-ra/ or /rena/is used when the possessor is inanimate, /-ala/ is used when the possessor is animate but the object that is possessed is inanimate, /-ali/ is used when the possessor as well as the possessed are both animate as shown below.

9. kitab **-ra/rena** panna kalɽʰae -me
   book **-GEN** page flip -IMP
   "Turn pages of the book"

10. ram **-ala** oɽaʔ
    Ram **-GEN** house
    "Ram's house."

11. ram **-ali** behen
    Ram **GEN** sister
    "Ram's sister "

Another kind of genitive marker is /t/ which is followed by the pronominal clitic, attaches to the possessor whether the possessed object is animate or animate.

12. mamu **t=iŋ** -ala oɽaʔ
    maternal uncle **GEN=1S** GEN house
    "My maternal uncle's house"

3.6 Infinitives

Asur does not have a marker for infinitives. The verb without any tense/aspect marker is an indicator that the verb is infinitive. The verb for 'sleep' i.e /nindao/ is in infinitive form in the example below.

13. iŋ nindao nanam-tan=iŋ
    1S sleep want-Prog=1S
    'I want to sleep. (I am wanting to sleep)'

3.7 Tense Aspect Mood

3.7.1 Tense

All languages have ways of expressing time. Tense and aspect markings in Asur cannot be generated in a formulaic manner; it is morphologically very complex and inconsistent like Mundari; as described by Osada (2008) and Langendoen (1963) .

[a] **Present tense:** In Asur language the present tense is unmarked or it may be said that it exists as null morpheme.

14. iŋ roj ul jom=iŋ
    1S everyday mango eat=1S
    "I eat mango every day."

[b] **Past tense:** In Asur past tense is marked by /l/, /ne/ and /ke/. Sentences (15) and (16) are past tense sentences in Asur.

15. iŋ ul jom-l=iŋ
    1S mango eat-PST=1S
    "I ate mango."

16. iŋ ul jom-ke-d=iŋ
    1S mango eat-PST-TR=1S
    "I ate mango."

[c] **Future tense:** The future tense marker in Asur is /-ke/ which is homophonous with the past tense /-ke/ marker as shown in (17).

17. iŋ gapa ul jom-ke=iŋ
    1S tomorrow mango eat-FUT=1S
    "I will eat mango tomorrow."

3.7.2 Aspect

Aspect gives information about the inside of the situation, or views the situation as a whole, or in phases, or informs whether the action was complete or not.

[a] **Progressive aspect**: Progressive aspect describes the internal structure of progression of a situation or event. In Asur /-tan/ is a progressive aspect marker for present tense, as in (13).

[b] **Perfect aspect:** Perfect aspect which, is also called complete aspect has three time points - the point of speech, the point of event and the reference time. The marker /-ta/ is used in Asur to mark perfect aspect.

18. iŋ ul jom-ta-d=iŋ
    1S mango eat-Perf-TR=1S
    "I have eaten the mango."

3.7.3 Mood

Mood expresses attitude; grammatical inflections are used to convey the attitude related to what is being said.

[a] **Imperative mood** performs the function of forming command, request or advice. In Asur /-e/ is the imperative marker.

19. sen-am-e
    go-2S[-HON]-IMP
    "Go [-HON]."

[b] **Potential mood:** This mood indicated the speakers' opinion about what he/she considers likely. In Asur the potential mood /hui/, appears after the verb as shown below.

20. iŋ hola ul jom-tahi-l=iŋ **hui**
    1S yesterday mango eat-Perf-PST=1S **POT**
    "I might have eaten mango yesterday."

3.8 Compound verb

Compound verb is a kind of complex verb in which both polar and vector components are verbs. In the following Asur example (28) the polar verb is on the left and vector verb is on the right and take all the inflections.

28. iŋ hoɽ=ku ke **raʔ** **agu=ku=iŋ**
    1 person=3 AC **invit** **bring=3P=1**
    S P C **e** **S**
    "I will call and bring people."

3.9 Question

In Asur language the polarity questions i.e. questions that evoke a yes-no response can be formulated by changing the intonation of a declarative type sentence or with the help of polar question particle (PQP) /ci/.

21. am buru sen=am **ci**
    2S forest go=2S **PQP**
    'Did you go to the forest?'

The basic interrogative particle (Q) forms in Asur are /oka/, /oke/ and /etan/, /eta/ which stand for "who" and "what" respectively.

22. **oke~oke** bajar sen-ne-n=akun
    Q~Q market Go-PST-ITR-P.INDF
    "Who all went to the market?"

23. am **etan** likkʰa-tan=am
    2S **Q** write-Prog=2S
    "What are you writing?"

In Asur /ikin/ is an interrogative particle which is used in relation to time.

24. **ikin** bajao-tana
    Q time-Prog
    "What is the time? (What time is happening?)"

3.10 Negation

Negation marker in Asur is /kae/, /kaeʔ/ or /ka/ and /ir/. Negation marker /kae/ or /ka/ can be roughly translated as "do not" as shown in (25) and /ir/ is used in the context of "will not" as shown in (26).

25. kami **ka** huiyo-n-a
    work **NEG** happen-ITR-IND
    "Work did not happen."

26. am **ir**=am jome=am
    2S **NEG**=2S eat=2S
    "You will not eat."

Example (27) shows another form of negative particle in Asur i.e. /kania/ which is mainly used to negate adjectives.

27. bes **kania** koɽa
good **NEG** boy
"not good boy"

3.11 Conjunction

Conjunction is a word or morpheme that conjoins or links words or larger constituents and also expresses some semantic relation between them. In Asur words /heʔ/ and /hetra/ are used as connective "and".

28. bariya pen **heʔ/hetra** miyaḍ kitab
two pen **and.CONJ** one book
"Two pens and one book."

In Asur /kowando/ can be translated as "either". The phrase /eta lae ci/ is used as connective "because" or "that is why" and the word /makil/ can be translated as "but" also functions as conjunction.

4 Other Typological features of Asur

Asur has basic word order SOV. The knowledge of constituent orderings in a language is essential to form grammatical utterances in that language.

[a] Adjective precedes the noun it modifies: In Asur adjective precedes the noun it modifies as shown in (29).

29. **bes** bilai
good.Adj cat
"Good cat"

[b] Relative clauses precede noun it modifies: In Asur language as well like in most OV languages the relative clause precedes the noun it modifies, as shown below:

30. **je** kitab am kiriŋ-l=am
REL book 2S buy-PST=2S
"The book that you bought..."

[c] Standard of comparison precedes the comparative adjective: Standard of comparison in Asur precedes the comparative adjective. In the following sentence /tara bes/ is the comparative adjective.

31. sila suman **tara** **bes** kuɽi hake
Seela Suman **ABL** **good** girl COP.PRS
"Sheela is better girl than Suman."

[d] Adverb precedes verb: In Asur language, adverbs precede verbs, as shown in example (32).

32. huni **muruk** **jorse** gul-l=ae
3S **very.Adv** **loudly.Adv** shout-PST=3S
"He shouted very loudly."

5 Borrowing

Most of the native speakers of Asur are multilingual. Asur speakers are in constant contact with other family of languages like Indo Aryan (Sadri and Hindi) and Dravidian (Kurux). Asur speakers frequently use not only Hindi and Sadri noun loan words, but also copular forms like /heke/ or /hake/ (Kavita Kiran). Many Hindi and Sadri verbs have been accommodated in Asur lexicon. The sound /-ao/ is added to the borrowed verb root to form Asur verb root or infinitive verb in Asur (Khalid, 2020). As Asur is in close proximity with other languages, the study of contact situation and borrowings from other languages into Asur would be interesting.

6 Conclusion

The paper is intended to put forth a brief sketch of Asur grammar; there is significant scope for further work on comprehensive grammar and lexicon for the documentation of Asur language. A comprehensive grammar and documentation work would open up possibilities for further research based on the existing Asur data and analysis. A conclusive work on sociolinguistic variations would require a larger number of native speakers from different age groups, gender and regions. There is scope for extensive study on linguistic variation of Asur language. There is an immense scope to investigate it historically in the context of proto Munda reconstruction, and also to conduct a comparative study of Asur with other neighboring languages and Munda languages.

Acknowledgement

This sketch grammar is an outcome of a UGC project entitled 'Establishment of Centre for Endangered Languages in Central Universities' ongoing in Central University of Jharkhand. My heartfelt gratitude goes to the native speakers of Asur, especially Mr. Jogeshwar Asur for his tireless cooperation. I would like to thank the project team members Mr.Gunjal Ikir Munda, Ms Shilpa, Ms. Rishika, Mr. Arun and also Mr.Sudhanshu Shekhar and Prof.Subbarao for their support.

References

Anderson, G. D. (2008). *The Munda Languages.* Routlege.

Fieldworks. (2020). Retrieved from SIL Language Software: https://software.sil.org/fieldworks/

Jenny, M., Weber, T., & Waymuth, R. (2015). 2 The Austroasiatic Languages: A Typological Overview. In *The Handbook of Austroasiatic Languages* (Vol. 2 vols, pp. 13-143). Brill.

Kavita Kiran, J. P. (n.d.). Sadani/Sadri.

Khalid, Z. (2020). A Phonological Sketch of Asur. *Language in India* .

Langendoen, D. T. (1963). Mundari Phonology. *unpublished paper* .

Munda, R. D. (1971). Aspects of Mundari Verb. (32).

Osada, T. (1992). *A reference grammar of Mundari.* Institute for the Study of Languages and Cultures of Asia and Africa, Tokyo University of Foreign Studies.

Osada, T. (2008). Mundari. In G. D. Anderson (Ed.), *The Munda Languages.* USA: Routledge.

SayMore. (2020). (SIL International) Retrieved from SIL Language Software-SIL International: https://software.sil.org/saymore/download/

Simons, Gary, F., & Charles, D. F. (2018). *Ethnologue*, Twenty first. (SIL International) Retrieved May 2, 2018, from Ethnologue: Languages of the World: http:/www.ethnologue.com

Subbārāo, K. V. (2012). *South Asian languages: a syntactic typology.* Cambridge University Press.

UNESCO. Director General,2009-2017(Bokova, I.G), writer of preface. (2010). *Atlas of the World's Languages in Danger.* (3. edn., Ed.) Paris: UNESCO Publishing.

Appendix A. Basic Lexical Items of Asur

Sl.no.	English Word	Asur translation
1.	I	/iŋ/
2.	you (singular)	/am/
3.	he	/ae/
4.	we	/abu/
5.	you (plural)	/am/
6.	they	/aku/
7.	this	/niya/
8.	that	/hona/
9.	here	/noaʔɽe/
10.	there	/honaʔɽe/
11.	who	/oke/
12.	what	/etan/
13.	where	/okaɽe/
14.	when	/okahila/
15.	how	/etalakan/
16.	not	/kae/
17.	all	/jamma/
18.	many	/jameku/
19.	some	/kaʈiʔ/
20.	few	/tʰoɽe/
21.	one	/miyaɖ/
22.	two	/bariya/
23.	three	/peya/
24.	four	/cair/
25.	five	/pac/
26.	big	/baɽe/
27.	long	/jiliŋ/
28.	wide	/cakeɽ/
29.	thick	/gaɽha/
30.	heavy	/bojʰ/
31.	small	/cʰoɽe/
32.	tall	/usul/
33.	short	/ʈʰepre/
34.	narrow	/sakuɽ/
35.	thin	/patla/
36.	girl	/kuɽi/
37.	boy	/koɽɑ/
38.	man (human being)	/hoɽ/
39.	child	/ceŋa/
40.	wife	/hoɽaʔ/

41.	husband	/hoɽ/
42.	mother	/aya/
43.	father	/baba/
44.	animal	/janwar/
45.	fish	/haku/
46.	bird	/oɽe/
47.	dog	/seta/
48.	deer	/saram/
49.	rabbit	/kulahi/
50.	goat	/merom/
51.	pig	/sukri/
52.	louse	/siku/
53.	snake	/biŋ/
54.	tree	/siŋ/
55.	forest	/buru/
56.	stick	/hapa/
57.	fruit	/joʔ/
58.	mango	/ul/
59.	seed	/bihin/
60.	leaf	/sekam/
61.	root	/jaiɽ/
62.	bark (of a tree)	/bakla/
63.	flower	/baha/
64.	grass	/gʰas/
65.	rope	/bayor/
66.	skin	/harta/
67.	meat	/sikar/
68.	blood	/mayom/
69.	bone	/jaŋ/
70.	fat (noun)	/itil/
71.	egg	/bili/
72.	horn	/diriŋ/
73.	tail	/calom/
74.	hair	/ub/
75.	head	/boho/
76.	ear	/lutur/
77.	eye	/meɖ/
78.	nose	/muhu/
79.	mouth	/aha/
80.	lips	/lucir/
81.	tooth	/ɖaɽa/
82.	tongue (organ)	/alaŋ/
83.	fingernail	/rama/
84.	leg	/jaŋga/
85.	knee	/mukɽi/
86.	hand	/ti/
87.	wing	/apaɽa/
88.	belly	/lahiʔg/
89.	guts	/poɽa/
90.	neck	/hoɽoʔ/
91.	back	/dea/
92.	breast	/cʰati/
93.	heart	/iyaʔ/
94.	liver	/karja/
95.	to drink	/nu/
96.	to eat	/jom/
97.	to bite	/hab/
98.	to suck	/cepeʔg/
99.	to spit	/beʔg/
100.	to vomit	/cʰaɽ/
101.	to blow	/om/
102.	to breathe	/sas/
103.	to laugh	/landa/
104.	to see	/nel/
105.	to hear	/ayum/
106.	to know	/paɽi/
107.	to think	/uihar/
108.	to fear	/bor/
109.	to sleep	/nind/
110.	to live	/jiuɖ/
111.	to die	/goeʔg/
112.	to fight	/jʰagɽa/
113.	to hit	/ɖʰesa/
114.	to cut	/geɖ/
115.	to split	/rupuɽ/
116.	to stab	/jobao/
117.	to scratch	/godar/
118.	to dig	/gotaʔ/
119.	to swim	/paurao/
120.	to fly	/otaŋ/
121.	to walk	/sen /
122.	to come	/hijoʔ/
123.	to lie (as in a bed)	/gitiʔg/
124.	to sit	/duɽuʔ /
125.	to turn (intransitive)	/muhaɖ/
126.	to fall	/uyuʔ/
127.	to catch	/sab/
128.	to squeeze	/cipuɖ/
129.	to wash	/dʰoao/
130.	to wipe	/joʔ/
131.	to push	/tukun/

132.	to tie	/tol/
133.	to sew	/roʔ/
134.	to count	/lekʰa/
135.	to say	/kahɽi/
136.	to sing	/siriŋ/
137.	to play	/eneʔg/
138.	to float	/capi/
139.	to flow	/tu/
140.	to freeze	/jamao/
141.	to swell	/mo/
142.	sun	/din boŋa/
143.	moon	/canu/
144.	star	/ipil/
145.	water	/daʔ/
146.	rain	/barkʰa/
147.	river	/nai/
148.	pond	/pokʰra/
149.	salt	/buluŋ/
150.	stone	/ɽuku/
151.	sand	/bitil/
152.	dust	/dʰuɽi/
153.	earth	/ot/
154.	cloud	/badri/
155.	fog	/dhʰudʰ/
156.	wind	/hoe/
157.	ice	/rataŋ/
158.	smoke	/sukul/
159.	fire	/seŋgel/
160.	ash	/toreʔg/
161.	to burn	/jul/
162.	road/path	/hora/
163.	mountain/hill	/buru/
164.	red	/lal/
165.	green	/hariar/
166.	yellow	/sasaŋ/
167.	white	/puɽi/
168.	black	/kaɽia/
169.	night	/nida/
170.	day	/din/
171.	year	/sal/
172.	today	/tisiŋ/
173.	tomorrow	/gapa/
174.	yesterday	/hola/
175.	warm	/lolo/
176.	cold	/tʰaɽ/
177.	full	/pereʔg/
178.	new	/nawa/
179.	old	/mari/
180.	good	/baɽʰiya/
181.	bad	/kʰarab/
182.	rotten	/sia/
183.	dirty	/gadus/
184.	straight	/sojʰ/
185.	round	/gutaɽ/
186.	square	/caukuɽ/
187.	sharp (as a knife)	/dʰar/
188.	dull (as a knife)	/bokʰa/
189.	wet	/lepa/
190.	dry	/rohoɽ/
191.	correct	/tʰik/
192.	near	/hinaɖ/
193.	far	/laŋka/
194.	right	/jom/
195.	left	/peŋka/
196.	at	/re/
197.	in	/re/
198.	with	/lo/
199.	and	/hetra/
200.	name	/numu/

Appendix B. Asur Phonology

Asur has consonant inventory similar to that of Mundari. The main difference is that Asur does not have voiced palatal nasal sound (Osada, 1992) , (Osada, 2008) like Mundari. Asur language has the following consonants: (Khalid, 2020)

Plosives: /p/, /b/, / t/, / d/, /ʈ/, /ɖ/, /k/, /g/. (/t/ and /d/ here are dental plosives.)

Like Mundari and few other Munda languages (Anderson, 2008) phoneme /g/ has an allophonic variation [ʔ] and glottal followed by unreleased /g/ sound, i.e [gˀ].

Aspirated plosives: /pʰ/, /bʰ/,/ t̪ʰ/, / d̪ʰ /, /ʈʰ/, /ɖʰ/, /kʰ/, /gʰ/

Nasals: /m/, /n/, /ŋ/

Trill: /r/

Flap: /ɽ/

Fricative: /s/, /h/,

Approximants: /w/, /y/

Lateral approximants: /l/

Affricates: /ʧ/, /ʤ/ (written as /c/ and /j/ in the peper)

Aspirated affricates: /ʧʰ/, /ʤʰ/

Vowels in Asur are as follows: (Khalid, 2020)

Rounded: /o/, /u/

Unrounded: /i/, /e/, /a/

There are five vowels in Asur which have several allophonic variants, like /e/ may appear as [ɛ], /a/ may appear as [ə], [æ] or[ɑ] and /o/ may appear as [ɔ]. Vowel length is not phonemic in Asur, i.e. the change in vowel length does not affect the meaning of the word in Asur.

In Asur nasalization is not phonemic. It occurs usually on the vowels preceding or following nasal consonants like Mundari (Osada, 2008). Nasalization sound is also often heard in onomatopoeic words like /cũ cũ/ "sound of rat", /kõ kõ/ "sound made by monkey" (Khalid, 2020).

Intra-syllabic consonant clusters are rarely seen in Asur.

Appendix C. Some features of Asur

[a] Gender

The feminine and masculine categories of some words in Asur are phonologically independent, such as -"cow" /gae/ and "ox" /urik/; "hen" /sim/ and "rooster" /kʰokhro/. Mostly there is a generic name for animals in Asur, but when the speaker needs to specify the gender of the animal the word /airra/ "male" or /eŋa/ "female" is used as an adjective.

Table A: Natural gender terms

Sl.no.	Words in English	Asur words	
		Masculine	Feminine
1.	boy/girl	/koɽa/	/kuɽi/
2.	monkey	/bandra/	/bandri/
3.	ass	/gadʰa/	/gadʰi/
4.	wild cat	/airra bʰa:ɽo/	/eŋa bʰaɽo/
5.	dog	/airra seta/	/eŋa seta/

Some typical Asur names have both male and female versions. The male name often ends with /a/ and the female names with /i/.

Table B: Male and female names of Asur people

Sl.No.	Male names	Female names
1.	/kandra/	/kandri/
2.	/lamta/	/lamti/
3.	/birsa/	/birsi/

Feminine occupational terms in Asur are derived from masculine occupational terms by suffixing /-in/, although occupational terms are mostly borrowed from Sadri. Occupation terms in Asur are shown in Table (C).

Table C: Occupational terms (masculine and feminine)

Sl.No	Masculine	Feminine
1.	/masʈar/ Teacher(M)	/masʈarin/ Teacher(F)
2.	/ɖraibʰar/ Driver(M)	/ɖraibʰarin/ Driver(F)

[b] Inclusive and Exclusive markers in Asur

Inclusive markers are markers which are used to include more to the noun phrase. It can be translated as "also" in English. Indo-Aryan languages very frequently have this marker. In Hindi /bʰi/ performs the function of inclusive marker. Sentence (a) shows inclusive marker /hoʔ/ in Asur.

i. iŋ **hoʔ** sen=iŋ
1S **INC** go=1S
"Shall I also go?"

Exclusive markers can be roughly translated as "only" or "just". Hindi also has Exclusive marker i.e /hi/. In (b) the the Asur exclusive marker /gi/ is used to emphasize the exclusivity of the subject.

b. iŋ **gi** ot sen=iŋ
1s **EXM** field go=1S
"Only I will go to the field."

[c] Transitivity marker

Asur marks intransitive sentences with /-n/ and transitive sentences with /-d/. Transitivity marker is a common feature of North Munda languages. This marker is also found in Mundari and has been termed "subject focus marker" and "object

focus marker" (Munda, 1971). These markers are not obligatorily present to mark transitivity and intransitivity in Asur language.

ii. abu ghoɽo jom-ke-**d**=abu
1P Food eat-Perf-**TR**=1P
"We ate food."

iii. iŋ hola nir-ke-**n**=iŋ
1S yesterday Run-PST-**ITR**=1S
"I was running yesterday."

[d] Causatives

The causativization marker /-ci/ in Asur, increases valency of the the verb by two, therefore this is the second causative marker.

iv. iŋ ceŋa ke towa uyuʔ-**ci**-l=iŋ
1S child ACC milk Fall-**CAUS**-PST=1S
"I made the child spill milk."

[e] Conjunctive Participle marker

A participle is a form of non-finite verb which acts as adjective or adverb. Conjunctive participles are present in most Indian languages, and perform various functions. They may act like time adverb, manner adverb, reason adverb; they may also join clauses and give the sense of sequential action. The conjunctive participle marker in Asur is /kʰete/ or /kʰe/ which can be used interchangeably, although /kʰete/ occurs more frequently.

v. iŋ ɖiɽoʔ **kʰeṭe** hiʔ
1S walk **CPM** come.
"We came walking."

English to Manipuri and Mizo Post-Editing Effort and its Impact on Low Resource Machine Translation

Loitongbam Sanayai Meetei[1], **Thoudam Doren Singh**[1], **Sivaji Bandyopadhyay**[1], **Mihaela Vela**[2], and **Josef van Genabith**[2,3]

[1]Centre for Natural Language Processing (CNLP) & Dept. of CSE, NIT Silchar, India
[2]Dept. of Language Science and Technology, Saarland University, Saarbrücken, Germany
[3]DFKI, Saarbrücken, Germany
{loisanayai,thoudam.doren,sivaji.cse.ju}@gmail.com
m.vela@mx.uni-saarland.de, josef.van_genabith@dfki.de

Abstract

We present the first study on the post-editing (PE) effort required to build a parallel dataset for English-Manipuri and English-Mizo, in the context of a project on creating data for machine translation (MT). English source text from a local daily newspaper are machine translated into Manipuri and Mizo using PBSMT systems built in-house. A Computer Assisted Translation (CAT) tool is used to record the time, keystroke and other indicators to measure PE effort in terms of temporal and technical effort. A positive correlation between the technical effort and the number of function words is seen for English-Manipuri and English-Mizo but a negative correlation between the technical effort and the number of noun words for English-Mizo. However, average time spent per token in PE English-Mizo text is negatively correlated with the temporal effort. The main reason for these results are due to (i) English and Mizo using the same script, while Manipuri uses a different script and (ii) the agglutinative nature of Manipuri. Further, we check the impact of training a MT system in an incremental approach, by including the post-edited dataset as additional training data. The result shows an increase in HBLEU of up to 4.6 for English-Manipuri.

1 Introduction

In our increasingly globalized world, communication plays a vital role and with it, demand for translation between different languages is on the rise. Despite much progress, machine translation (MT) on its own may not always be sufficient to meet the demand. MT output may sometimes be erroneous and needs to be checked and corrected. The use of translation technology such as MT systems, translation memories (TM) and CAT tools can boost translation productivity (Koehn, 2009; Plitt and Masselot, 2010). However, limited numbers of professional translators for a language pair can be a major challenge, especially for low resource languages.

Raw MT output is not always exempt from errors. Often post-editing MT output (where a human translator reviews and where required corrects MT output) is the most productive approach to translation. PE effort is the amount of effort required to generate a reasonable target text from MT output. Following Krings (2001) PE effort can be subdivided into **temporal effort**, **technical effort** and **cognitive effort**. Temporal effort represents the overall time taken to complete a PE task. Technical effort can be measured tracking keyboard and mouse interactions, including insertion, deletion, mouse movement, etc. Cognitive effort (considered the most difficult to measure) involves mental effort such as reading and understanding the text, identifying errors, and the decision making process towards correcting errors.

To date, PE research has mainly concentrated on a few well-studied languages. In this work, the same source data in English are machine translated to low resource languages to carry out a PE task. The dataset used in the experiment consists of news articles collected from a local daily newspaper, Imphal Free Press[1] in Manipur, a Northeastern state of India with a population of around 3 million[2] and a geographic size of 22,327 sq. km. The collected news corpus is originally in English and then machine translated into three

[1]https://ifp.co.in/
[2]http://censusindia.gov.in

Proceedings of the 17th International Conference on Natural Language Processing, pages 50–59
Patna, India, December 18 - 21, 2020.

English	Manipuri	Mizo	Hindi
SVO	SOV	OSV	SOV
Roman	Bengali	Roman	Devanagari

Table 1: Typological Word Order and Script of Languages in the Study.

Acronyms: O = Object, S = Subject, V = Verb.

target languages Manipuri, Mizo and Hindi resulting in three parallel datasets. The basic word order of the languages involved in our experiment along with their scripts are listed in Table 1.

We conduct a study on the PE effort required to produce reasonable target text in English-Manipuri and English-Mizo. As there is no commercially available machine translation system for these two languages, for comparison we also studied English-Hindi PE effort on commercial MT output on the same dataset. Two levels of PE are generally distinguished: light and full. For our experiments, we instruct our post-editors to carry out light PE to achieve the desired level of output quality. With various PE effort indicators computed using the data captured from the CAT tool, we carry out an experiment to measure the PE effort for English-Manipuri, English-Mizo and English-Hindi MT systems. Lexical words and function words are observed to have a different impact on the PE effort on the MT output for the three language pairs. We also carry out an experiment to test the impact of training a machine translation system for the low resource language pairs English-Manipuri and English-Mizo in an incremental manner, i.e. by adding the PEed dataset to the original MT training data. The rest of the paper is structured as follows: Section 2 reviews previous research, Section 3 desrcibes the PE task and the human PEers, Section 4 presents our approach and system set up. Section 5 details our findings. Section 6 summarizes our main results and avenues for further research.

2 Related Work

Early studies on the correlation between PE effort and various aspects of PE include O'Brien (2005). O'Brien studied the temporal, technical and cognitive effort involved in PE by analyzing keyboard-data using Translog and Choice Network Analysis (CNA). Several studies investigated bi-lingual PE and monolingual PE. In bilingual PE (Zampieri and Vela, 2014) post editors have access to the source text, while in monolingual PE (Nitzke, 2016) MT output is edited without the source text. Zampieri and Vela (2014) studied the use of TMs generated by MT output and their effect on human translation. The authors reported a significant increase in translation speed while using the TM as compared to translating from the scratch.

Similarly, Toral et al. (2018) show that post-editing an MT dataset involves less effort than translating from scratch. Post-editing MT output increases the productivity of the translators. Zaretskaya et al. (2016) examine various types of MT errors and the challenges they present for PE. Burchardt et al. (2013) compile a corpus consisting of English to German translation generated by different types of MT systems. The dataset is then annotated for translation errors using the MQM error typology, with only one error in each sentence. As the dataset is already annotated, the post-editors could skip the effort of identifying the errors and concentrate only on the highlighted error text segment in the PE process. Focusing on how PE effort changes with the different types of MT errors, the authors reported a weak correlation between PE time and PE effort. The authors also report that no direct dependency was found between the temporal and technical PE effort. Investigating the various types of PE operations for French to English and English to Spanish translation outputs, Popovic et al. (2014) reported lexical edits as the main factor in PE time.

Koponen et al. (2012), study the cognitive effort of post-editing MT output based on measuring PE time and HTER (Snover et al., 2006). HTER (Human-targeted Translation Edit Rate), is an automatic metric that computes the minimum number of edits required to change MT output into the post-edited version. The authors reported that the absolute PE time increases with the number of printable keystrokes and sentence length while seconds per word remain relatively constant. Despite the fact that HTER captures the difference between the final translation and raw

MT, it does not disclose much of the time and keystroke effort required to produce the final result. A similar study is also reported by Moorkens et al. (2015) where the human (or H-) variants of the reference based similarity measure such as BLEU (Papineni et al., 2002) is used to analyze PE effort.

Singh and Bandyopadhyay (2010a); Singh (2013) focus on MT for English to Manipuri, Pathak et al. (2019) on English to Mizo and Singh et al. (2017); Meetei et al. (2019b) on English to Hindi, using different MT approaches. But, to date there is no report on PE effort required to turn raw MT output for these target languages into useful translations. To address this gap in the literature, our paper investigates different aspects that impact PE effort and time spend to generate a reasonable target text from MT into Manipuri, Mizo and Hindi.

3 Description of the PE Task

Two post-editors who are native speakers of the target languages and also proficient with the source language are employed for each of the language pairs to carry out the PE task. For English-Manipuri and English-Hindi, the post-editors are undergraduate students of Computer Science and Engineering and for English-Mizo, the post-editors are postgraduate students of Science. When PEing machine translated text, it is important to clearly define what level of output quality is to be achieved. Generally two PE levels are distinguished: light or complete. In our work, the post-editors are asked to carry out light PE with the following instructions: 1) Using the maximum possible amount of raw MT text in the output of PE. 2) Ensure no addition or omission of source content. 3) Restructuring output, where the meaning is inaccurate.

4 Methodology and Experimental Design

We use an English language corpus collected from a local daily newspaper as the source text. We normalize the data in a pre-processing step. The normalized text is then machine translated into three different target languages using different MT systems. After post-editing a sample dataset of the machine translated text using a CAT tool, we study the data collected from the CAT tool to analyze PE effort and the time required to generate a reasonable target text. A pictorial representation of our experimental design is shown in Figure 1. The remainder of this section details individual steps in our approach.

	Sentences	Tokens
Total Dataset, D	64976	1688440
Sample Dataset, D_{PE}	200	5500

Table 2: Statistics of our collected dataset and data partitioning.

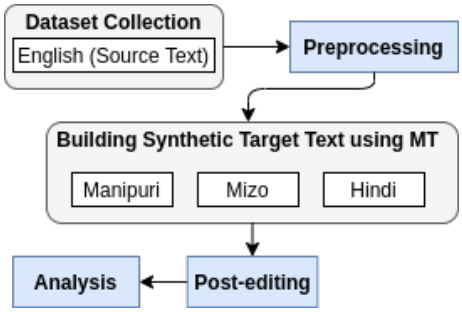


Figure 1: Experimental design.

4.1 Data Collection

The dataset used in our experiment is collected from a local daily newspaper based in Manipur, Imphal Free Press[3]. The news articles are in English. The complete dataset consists of 3770 news articles from the period July 2011 to October 2019 comprising 64976 sentences. We randomly select 200 sentences (D_{PE}) for our PE experiment. The statistics of the dataset and data partitioning are shown in Table 2. The dataset is collected using a web-scrapper built in-house.

4.2 Pre-processing

Data collected from the web is not free from noise. The pre-processing step includes removal of non-ascii special characters. Each of the news articles in our dataset is split into sentences using the Moses tokenizer (Koehn et al., 2007).

4.3 Building Machine Translated Target Text

We build a machine translated dataset using MT systems resulting in three language pairs,

[3]https://ifp.co.in/

	Sentences	Tokens	Types
en-mn	18070	*en*:390141; *mn*:358947	*en*:27891; *mn*:54611
$mono_{mn}$	131755	2798317	270998
en-mz	7500	*en*:86353; *mz*:87511	*en*:4301; *mz*:6151
$mono_{mz}$	1005675	29338218	312062

Table 3: Dataset for PBSMT systems.
en : English, *mn* : Manipuri, *mz* : Mizo.

namely, English-Manipuri, English-Mizo and English-Hindi.

4.3.1 English to Manipuri and Mizo MT

Manipuri and Mizo are the lingua francas of Manipur and Mizoram, two neighbouring north-eastern states of India. Both Manipuri and Mizo are low resource languages. Limited availability of data in Manipuri and Mizo is one of the main reasons that hamper the development of NLP systems for the language. The training datasets used for training the MT system for the languages are shown in Table 3. On the same English-Manipuri training dataset, we first examine the performance of MT systems trained with Phrase Based Statistical Machine Translation, PBSMT (Koehn et al., 2003) and the RNN-based NMT with attention mechanism (Bahdanau et al., 2014). The trained MT systems are evaluated on a held-out test dataset of 900 sentences. The result shows a BLEU score of 6.45, (34.7/9.6/3.5/1.5) on the PBSMT system while the NMT system achieved a BLEU score of 0.00, (11.8/0.3/0.0/0.0). For this reason, we use PBSMT systems for both English-Manipuri and English-Mizo MT systems as our parallel NMT results are substantially worse in these low-resource scenarios. To build language models for the target languages, we used the dataset in (Singh and Bandyopadhyay, 2010b) and (Meetei et al., 2019a) for Manipuri and Mizo respectively. mgiza[4] is used to generate the phrase table and srilm[5] to build the language model.

[4]https://github.com/moses-smt/mgiza
[5]http://www.speech.sri.com/projects/srilm/

4.3.2 English to Hindi MT

In order to translate the English dataset to Hindi, we use Google Translate which is a Neural Machine Translation (NMT) system.

4.4 Post-editing

To investigate PE effort, we randomly select a subset of 200 sentences from the original English data and automatically translate it into the three target languages. We create a translation memory (TM) for each of the language pairs to prepare the source and MT output data for use with a CAT tool. The resulting TMs are uploaded in a commercial CAT tool[6].

4.4.1 PE effort indicators

During the post-editing process using the CAT tool, we record post-editing logs capturing Seconds per Word, Time to Edit and Post-editing Effort for each sentence. We measure:

1. *Post Editing Time (PET)*: Total time taken to edit a sentence in the target language.
2. *Post Editing Effort*[6] *(PEE)*: Post-editing effort expended on the machine translated output to produce the desired target text per sentence. *PEE* is computed based on edit distance measured in words obtained using a heavily customized version of the Levenshtein distance algorithm (Levenshtein, 1966).
3. *Seconds per Word (SpW)*: The *PET* spent by the translator to post-edit divided by the number of tokens of the post-edited translation.
4. *Total number of tokens (TT)*: Total number of tokens per sentence in the source language.
5. *Noun Words (NN)*: The word content that can be used to refer to a named entity, quality or action.
6. *Lexical Words (LW)*: Lexical words per sentence in the source language. Lexical words are the essential building blocks of a language's vocabulary. Lexical words are nouns, adjectives, verbs, and adverbs.

[6]https://www.matecat.com

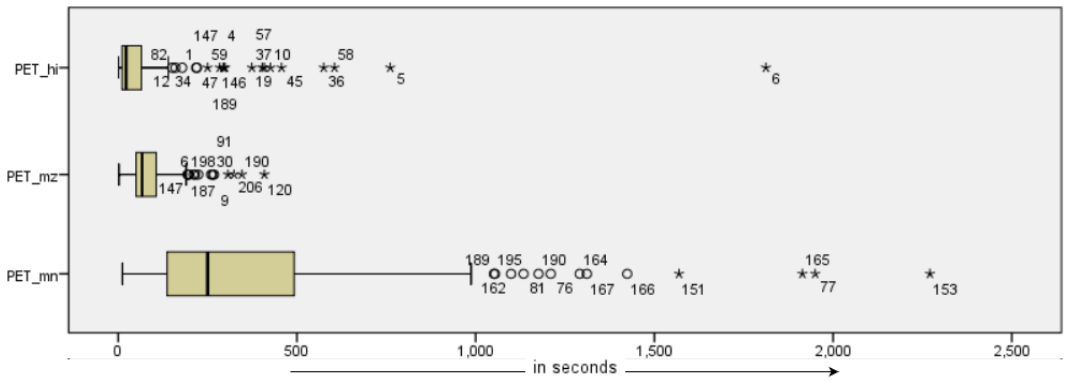


Figure 2: Distribution of Post-Editing Time (*PET*) for Manipuri, Mizo and Hindi

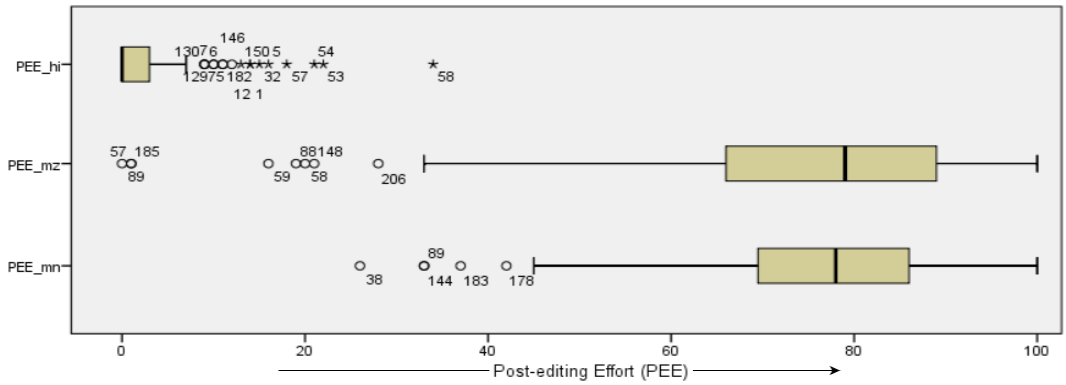


Figure 3: Distribution of Post-Editing Effort (*PEE*) for Manipuri, Mizo and Hindi

7. *Function words (FT)*: Function words per sentence in the source language. Function words are those words which are more grammatical in nature, such as articles, prepositions, etc. Here, $FT = TT\text{-}LW$.

Temporal effort is measured by the overall time taken *PET* while *PEE* represents the technical effort.

4.4.2 Descriptive Statistics and Correlation

We use mean, standard deviation as well as box plots to capture descriptive statistics of our datasets and results. To investigate whether variables co-vary we measure the correlation coefficient r with value between -1 to 1. A positive correlation shows the degree to which variables increase or decrease in parallel, while a negative correlation indicates that one variable increases as the other decreases.

5 Results and Discussion

To measure PE effort, PE logs are collected from the CAT tool after post-editing the raw MT output (in Manipuri, Mizo and Hindi) of the sample dataset, D_{PE}. We compare the general distribution of *PET* and *PEE* for each of the 200 sentences in the language pairs in the form of box plots. Mean and standard deviations for the rest of the source PE effort indicators (*TT, NN, LW, FT*) and the target PE effort indicators (*PET, PEE, SpW*) are computed and we investigate correlations between indicator variables.

5.1 Statistics and Correlation

Figure 2 and 3 show the distribution of post editing time (*PET*) and post editing effort (*PEE*) for the language pairs investigated. The box plots show:

- the minimum value, maximum value, first, second, third and fourth quartile of the experimental measures. The thick line represents the median.
- *Outliers*: these are the values that lie beyond the whiskers of a box plot. Outliers are marked by circles or asterisks along with their observation number. A circle represents an outlier (a value that appears to be outside of what is expected

for the observations), while asterisks represent extreme outliers (a value which is far away from what is expected).

Figure 2 shows that the *PET* of Manipuri (PET_mn) deviates far from the *PET* of Mizo (PET_mz) and Hindi (PET_hi). A likely cause of this is the massive amount of post-editing required in the output from English-Manipuri MT system combined with time spend on typing the Bengali script on the keyboard. Bengali scripts are used mostly by the news reporters while for daily communication, Roman scripts are used. To verify our findings, the English-Manipuri post-editors typed a set of randomly selected 50 English-Manipuri parallel sentences from the training corpus to measure the typing speed of Roman and Bengali scripts. 3001 seconds are spend on typing the English text consisting of 1153 tokens resulting in an average typing speed of 2.6 seconds per token. While for the Manipuri text with 1148 tokens, 5610 seconds are spend on typing, resulting in an average typing speed of 4.8 seconds per token. This led our post-editors to spend more time while post-editing a large portion of the translated text. The post-editing effort (*PEE*) for the three languages are shown in Figure 3. While the PEE for Hindi (PEE_hi) is small, the *PEE* of Manipuri (PEE_mn) and Mizo (PEE_mz) are very large with a maximum value of 100. This shows that massive effort is required in the post-editing task of Manipuri and Mizo resulting from low performance of the current state of the art of English-Manipuri and English-Mizo MT systems.

Mean and Standard Deviation (*SD*) of the indicators in the source language (English) of the dataset D_{PE} are shown in Table 4. To identify the lexical words (*LW*), we POS-tag the data using the Stanford Log-linear Part-Of-Speech Tagger[7].

Table 5 summarizes the the descriptive statistics, mean and SD of the indicators (*PET, PEE, SpW*) for our Manipuri, Mizo and Hindi experiments. Compared to Mizo and Hindi, *SpW* for Manipuri is longer. The main reason for this is the large portion of text post-edited and because of the difficulty in writing Bengali characters by the post-editors. The

[7]https://nlp.stanford.edu

	NN	*LW*	*FT*	*TT*
Mean	9.64	15.39	9.16	24.56
SD	6.06	7.51	5.28	11.85

Table 4: Descriptive Statistics of the source text of dataset D_{PE}.

Acronyms: NN: Nouns, *LW*: Lexical Words, *FT*: Function words, *TT*: Total tokens

	Mean	*SD*
PEE_{mn}	**76.35**	13.45
PEE_{mz}	75.02	19.22
PEE_{hi}	2.29	4.64
PET_{mn}	**400.78**	378.93
PET_{mz}	91.00	63.75
PET_{hi}	72.48	168.14
SpW_{mn}	**16.14**	13.55
SpW_{mz}	3.51	2.57
SpW_{hi}	2.93	6.14

Table 5: Descriptive Statistics of target text.

Subscripts- $_{mn}$: Manipuri, $_{mz}$: Mizo, $_{hi}$: Hindi.

result in Table 5 shows a mean value ≈ 76.35 and ≈ 75.02 in the *PEE* for English-Manipuri and English-Mizo dataset respectively. Much of this is due to the current state of the art of English-Manipuri and English-Mizo MT systems. We note that significant effort is required to improve the English-Manipuri and English-Mizo MT system which requires a $PEE > 55$ in all the cases.

5.2 Correlation

In our experiment, the correlation between PE effort indicators is computed using Pearson's correlation coefficient to determine whether there is a potential dependency between them. The correlations among the indicators for the three language pairs involved in our experiment are shown in Table 6.

A Pearson's r data analysis shows a significant positive correlation between PEE_{mn} and *FT* ($p<0.05$, $r= .16$) and also between PEE_{mz} and *FT* ($p<0.01$, $r= .27$). The result also shows a significant ($p<0.01$) negative correlation between PEE_{mz} and *NN*. The main reason for the above result is because Mizo uses the same script as the source text and the agglutinative nature of the Manipuri text. In Manipuri function words such as articles (a, the), prepositions (on, at, in), etc. are suf-

	NN	LW	FT	TT
PEE_{mn}	0.041	0.088	0.162*	0.128
PEE_{mz}	-0.300†	-0.066	0.274†	0.080
PEE_{hi}	0.049	-0.065	-0.047	-0.062
PET_{mn}	0.344†	0.407†	0.405†	0.438†
PET_{mz}	0.620†	0.646†	0.452†	0.611†
PET_{hi}	0.234†	0.213†	0.256†	0.249†
SpW_{mn}	-0.038	-0.067	-0.043	-0.062
SpW_{mz}	-0.100	-0.202†	-0.322†	-0.271†
SpW_{hi}	-0.040	-0.106	-0.105	-0.114

Table 6: Correlation of source text and target text indicators. ***Note:*** † significant at 0.01 level of significance. * significant at 0.05 level of significance.

HBLEU	1-g	2-g	3-g	4-g	Average
MT_{mn}	33.8	10.0	4.0	2.0	7.16
MT_{mz}	34.8	10.7	5.5	4.0	9.48
MT_{hi}	96.5	94.0	91.5	89.2	92.78

Table 7: Evaluation against post-edited dataset D_{PE}.

fixed to the noun words in most of the cases, resulting in the formation of a new word.

The PET for all the language pairs involved is observed to be positively correlated with all the source text indicators.

In terms of seconds per word, SpW, only the English-Mizo pair is significantly negatively correlated with LW, FT and TT. With an increase in the number of tokens in the source text, the average time spent per token decreases. A likely cause is the use of same script.

In addition to the computation of correlations between indicator variables, we also calculated automatic MT evaluation (4-gram HBLEU) scores between the raw MT outputs and their post-edited versions of dataset D_{PE} for each language pair as shown in Table 7.

5.3 A Control Experiment

As, to the best of our knowledge, this is the first paper to report on PE research on Manipuri, Mizo and Hindi, it is not clear how the results obtained compare with previous research on well-resourced languages. Furthermore, as the PE for Manipuri, Mizo and Hindi did not involve professional translators, but students who are native speakers of our target languages with excellent command of English, we conducted a control PE experiment with German as target language and professional translator trainees at Saarland University. Our data consists of the same dataset selected for the Manipuri, Mizo and Hindi experiments, translated into German by DeepL[8], and PEed by seven Translation Study MA students with native German from the English to German translation track of the degree. We used the same CAT tool as in our Manipuri, Mizo and Hindi experiments and collected the same set of measurements.

For English-German we measure mean values of 48.94 in PET and 7.14 in PEE, compared to 72.48 and 2.30 for Hindi (see Figures 2 and 3). As both German and Hindi are well supported languages (both Google Translate and DeepL are some of the strongest performing systems for the EN-Hi and EN-DE language pairs), this provides additional support that the Hindi PE results we report are reliable and properly indicative of the task. Further, and in turn, this "anchoring" of the Hindi PE results through the German PE results, supports our belief that the large gap between the Hindi and with that of Manipuri and Mizo results observed in our experiments is also reliable, and can be traced to the fact that Manipuri and Mizo are much less well supported by language technologies and data (here machine translation) than Hindi or German.

5.4 Training English to Manipuri and Mizo MT systems on additional PEed data

Further, in an effort to improve the English-Manipuri and English-Mizo MT systems, we train the PBSMT systems for the language pairs in an incremental approach. We use the PEed dataset of (D_{PE}) for each language pair as additional training data to retrain our PBSMT systems (MT-I_{mn} and MT-I_{mz}). We further increase the additional training data of English-Manipuri [$D_{PEed\text{-}2}$ = 200 (D_{PE}) + 656] to retrain English-Manipuri PBSMT system (MT-$I2_{mn}$) but could not acquire the same for English-Mizo due to the lack of post-editors. In order to check the quality improvement in the translated text, we compare the

[8]https://www.deepl.com/

	Sentences	Tokens	Unique tokens
$D_{PEed\text{-}2}$	856	20309	4884
D_{Ev}	50	434	293

Table 8: Dataset to retrain and evaluate MT systems. [D_{Ev}: Evaluation Dataset]

HBLEU	1-g	2-g	3-g	4-g	Average
MT_{mn}	21.9	4.6	0.6	0.3	2.14
$MT\text{-}I_{mn}$	22.6	5.0	0.9	0.4	2.45
$MT\text{-}I2_{mn}$	30.0	9.1	3.8	2.0	**6.78**
MT_{mz}	47.6	20.3	9.0	3.4	11.83
$MT\text{-}I_{mz}$	49.2	21.6	9.5	3.7	**12.64**

Table 9: Evaluation for English-Manipuri ($_{mn}$) and English-Mizo ($_{mz}$) MT systems on D_{Ev}.

	Sentence
Source	my stint as dc of tamenglong has been professionally and personally satisfying : armstrong pame.
MT_{mn}	ঐগী ওইনা stint মীৎয়েং থম্না অমনি ওফ tamenglong অসি professionally লৈ অমসুং personally armstrong pame satisfying :
$MT\text{-}I_{mn}$	ঐগী ওইনা stint দিসি ওফ tamenglong অসি professionally লৈ অমসুং personally armstrong pame মফমনি ।
$MT\text{-}I2_{mn}$	ঐগী stint তমেংলোং ডিষ্ট্রিক্টকি ডিসি ওইনা অসি লৈ অমসুং ইশাগি ওইনা মফমনি : অরমসত্রোং পামে
Reference	ঐনা মতম খরা তমেংলোংগী ডিসি ওইবসি শিনফমগী ওইনা অমসুং ইশাগী ওইনা অপেনবা ফাওই : অরমসত্রোং পামে
MT_{mz}	my stint , dc te chuan tamenglong bana professionally leh personally satisfying : armstrong pame
$MT\text{-}I_{mz}$	my stint dc te an nei a , tamenglong professionally leh personally satisfying : armstrong pame
Reference	tamenglong dc ka nih chhung hian hnathawh dan leh mimal tak pawhin hlawkna tam tak ka hmu : armstrong pame

Table 10: Sample Output of PBSMT systems.

HBLEU scores of the retrained PBSMT systems and the original PBSMT systems (MT_{mn} and MT_{mz}). Table 8 and 9 summarize the dataset used to evaluate the MT systems and their evaluations in terms of *HBLEU* score. Table 9 shows that the retrained MT systems gives clearly better results than original MT systems with an increase in *HBLEU* score of up to 4.6. Sample outputs from the MT systems are shown in Table 10.

6 Conclusions

Using log-information gathered from our CAT tool, an analysis of the PE effort and PE time is carried out for three target languages: Manipuri, Mizo and Hindi with English as the source language. To our knowledge, this is the first PE analysis conducted on English-Manipuri, English-Mizo and English-Hindi.

Our analysis shows that current state of the art in commercially available MT for English-Hindi requires small PE effort and PE time. While MT systems for low resource languages such as Manipuri and Mizo are under development, MT training data for the languages is very scarce. Using a PBSMT system built in-house, a study on the PE effort and PE time is carried out for English-Manipuri and English-Mizo. Our findings show that, compared to English-Manipuri and English-Mizo, *PEE* is low for English-Hindi. By contrast, for English-Manipuri and English-Mizo, the problems in MT output are far more serious requiring heavy PE effort. Interestingly, while there is a significant correlation between *PEE* and

FT for the language pair English-Manipuri and English-Mizo, there is a significant negative correlation between *PEE* and *NN* for the English-Mizo language pair. The *PEE* for English-Mizo decreases with the increase in noun words in the source text, which might be because of Mizo sharing the same script as the source language. Also, a significant negative correlation is observed between SpW and TT for English-Mizo. This suggests that with the increase in the number of tokens in source text, the average time taken per word decreases for English-Mizo. We identify MT quality as well as script and ease of typing script as a factor in PE effort for languages like Manipuri and Mizo.

We also made a first attempt to address the scarcity of a parallel training data of English-Manipuri and English-Mizo MT by training the MT systems in an incremental manner using additional data created by the PE experiment. The result indicates an improvement of up to 4.6 in *HBLEU* for English-Manipuri.

7 Acknowledgments

This work is supported by Scheme for Promotion of Academic and Research Collaboration (SPARC) Project Code: P995 of No: SPARC/2018-2019/119/SL(IN) under MHRD, Govt of India.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Aljoscha Burchardt, Arle Lommel, and Maja Popovic. 2013. Tq error corpus. Technical report, Technical Report Deliverable D 1.2. 1, QT Launchpad Project.

Philipp Koehn. 2009. A process study of computer-aided translation. *Machine Translation*, 23(4):241–263.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, pages 177–180.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Maarit Koponen, Wilker Aziz, Luciana Ramos, and Lucia Specia. 2012. Post-editing time as a measure of cognitive effort. *Proceedings of WPTP*, pages 11–20.

Hans P Krings. 2001. *Repairing texts: empirical investigations of machine translation post-editing processes*, volume 5. Kent State University Press.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Loitongbam Sanayai Meetei, Thoudam Doren Singh, and Sivaji Bandyopadhyay. 2019a. Extraction and identification of manipuri and mizo texts from scene and document images. In *International Conference on Pattern Recognition and Machine Intelligence*, pages 405–414. Springer.

Loitongbam Sanayai Meetei, Thoudam Doren Singh, and Sivaji Bandyopadhyay. 2019b. Wat2019: English-hindi translation on hindi visual genome dataset. In *Proceedings of the 6th Workshop on Asian Translation*, pages 181–188.

Joss Moorkens, Sharon O'brien, Igor AL Da Silva, Norma B de Lima Fonseca, and Fabio Alves. 2015. Correlations of perceived post-editing effort with measurements of actual effort. *Machine Translation*, 29(3-4):267–284.

Jean Nitzke. 2016. Monolingual post-editing: An exploratory study on research behaviour and target text quality. *Eyetracking and applied linguistics*, 2:83–108.

Sharon O'Brien. 2005. Methodologies for measuring the correlations between post-editing effort and machine translatability. *Machine translation*, 19(1):37–58.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Amarnath Pathak, Partha Pakray, and Jereemi Bentham. 2019. English–mizo machine translation using neural and statistical approaches. *Neural Computing and Applications*, 31(11):7615–7631.

Mirko Plitt and François Masselot. 2010. A productivity test of statistical machine translation post-editing in a typical localisation context. *The Prague bulletin of mathematical linguistics*, 93(1):7–16.

Maja Popovic, Arle Lommel, Aljoscha Burchardt, Eleftherios Avramidis, and Hans Uszkoreit. 2014. Relations between different types of post-editing operations, cognitive effort and temporal effort. In *Proceedings of the 17th annual conference of the european association for machine translation*, pages 191–198. European Association for Machine Translation Dubrovnik, Croatia.

Sandhya Singh, Ritesh Panjwani, Anoop Kunchukuttan, and Pushpak Bhattacharyya. 2017. Comparing recurrent and convolutional architectures for english-hindi neural machine translation. In *Proceedings of the 4th Workshop on Asian Translation (WAT2017)*, pages 167–170.

Thoudam Doren Singh. 2013. Taste of two different flavours: Which manipuri script works better for english-manipuri language pair smt systems? In *Proceedings of the Seventh Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 11–18.

Thoudam Doren Singh and Sivaji Bandyopadhyay. 2010a. Manipuri-english bidirectional statistical machine translation systems using morphology and dependency relations. In *Proceedings of the 4th Workshop on Syntax and Structure in Statistical Translation*, pages 83–91.

Thoudam Doren Singh and Sivaji Bandyopadhyay. 2010b. Web based manipuri corpus for multiword ner and reduplicated mwes identification using svm. In *Proceedings of the 1st Workshop on South and Southeast Asian Natural Language Processing*, pages 35–42.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, volume 200.

Antonio Toral, Martijn Wieling, and Andy Way. 2018. Post-editing effort of a novel with statistical and neural machine translation. *Frontiers in Digital Humanities*, 5:9.

Marcos Zampieri and Mihaela Vela. 2014. Quantifying the influence of mt output in the translators' performance: A case study in technical translation. In *Proceedings of the EACL 2014 Workshop on Humans and Computer-assisted Translation*, pages 93–98.

Anna Zaretskaya, Mihaela Vela, Gloria Corpas Pastor, and Miriam Seghiri. 2016. Measuring post-editing time and effort for different types of machine translation errors.

Learning to Interact: An Adaptive Interaction Framework for Knowledge Graph Embeddings

Chandrahas[1] **Nilesh Agrawal**[2*] **Partha Talukdar**[1]
[1]Indian Institute of Science, Bangalore, [2]Cohesity, Bangalore
(chandrahas, anilesh, ppt)@iisc.ac.in

Abstract

Knowledge Graph (KG) Embedding methods have been widely studied in the past few years and many methods have been proposed. These methods represent entities and relations in the KG as vectors in a vector space, trained to distinguish correct edges from the incorrect ones. For this distinction, simple functions of vectors' dimensions, called interactions, are used. These interactions are used to calculate the candidate tail entity vector which is matched against all entities in the KG. However, for most of the existing methods, these interactions are fixed and manually specified. In this work, we propose an automated framework for discovering the interactions while training the KG Embeddings. The proposed method learns relevant interactions along with other parameters during training, allowing it to adapt to different datasets. Many of the existing methods can be seen as special cases of the proposed framework. We demonstrate the effectiveness of the proposed method on link prediction task by extensive experiments on multiple benchmark datasets.

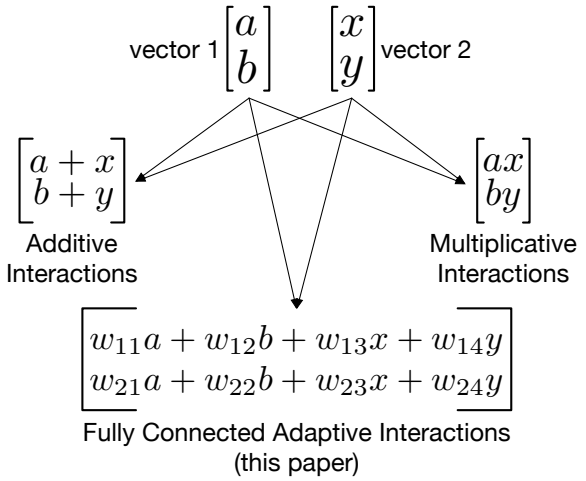


Figure 1: Some examples of interactions between two vectors. While additive and multiplicative interactions depend only on the input vectors, the fully connected (FC) interactions have trainable weights, allowing it to adapt to different datasets. In this example, additive interaction $a + x$ can be achived by FC interaction by setting $w_{11} = w_{13} = 1$ and $w_{12} = w_{14} = 0$. Similarly, the multiplicative interaction ax can be achieved by setting $w_{11} = 0.5x$ and $w_{13} = 0.5a$ and other weights as zero.

1 Introduction

Knowledge Graphs (KGs) such as NELL (Mitchell et al., 2015), Freebase (Bollacker et al., 2008), YAGO (Suchanek et al., 2007), etc. have been very popular in supporting many AI applications like Web Search Query Recommendation (Huang et al., 2016), Question Answering (Yao and Van Durme, 2014), Visual Question Answering (Shah et al., 2019) etc. KGs are multi-relational graphs containing entities as nodes and typed relations between entity pairs as edges. These graphs store real-world facts such as *(Lionel Messi, plays-for-team, Argentina National Football Team)* as edges, also called triples. In spite of their popularity, they suffer from incompleteness (Dong et al., 2014), and it becomes important to predict the missing edges in the graph. The task of predicting missing edges in a KG is called link prediction.

Knowledge Graph Embedding (KGE) methods have been a popular approach for the link prediction task. Most of these methods learn vectorial representations for entities and relations in the KG. A score function is then used to distinguish correct triples from the incorrect ones. Given a triple of the form (h, r, t) where h, r and t are the head entity, relation, and the tail entity, a score function assigns a real-valued score to the triple. These score functions depend upon the interactions of dimensions

*Work done while at the Indian Institute of Science, Bangalore.

Proceedings of the 17th International Conference on Natural Language Processing, pages 60–69
Patna, India, December 18 - 21, 2020.

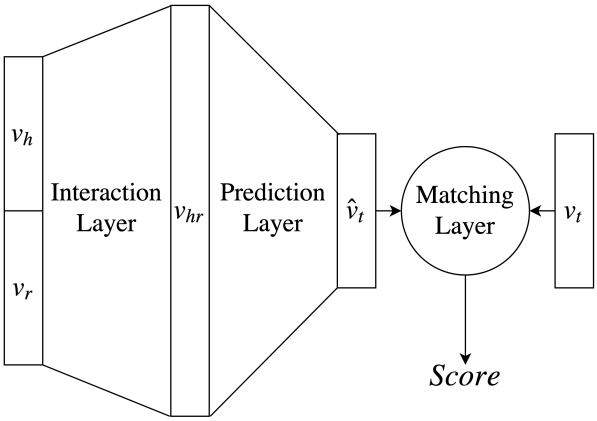


Figure 2: The block diagram of the Adaptive Interaction Framework. The interaction layer extracts interactions v_{hr} from head entity v_h and relation vector v_r. The prediction layer calculates a candidate tail entity vector $\hat{v}_t$ which is then matched with existing tail entity vector v_t using the matching layer to produce a real valued score.

of vectors for h and r. Some examples of interactions are given in Figure 1. TransE (Bordes et al., 2013) uses the additive interactions while DistMult (Yang et al., 2014) uses the multiplicative interactions. However, these interactions are fixed for a given method and are not learnable. This restricts models' capability to weigh entities and relations differently, and hence, from adapting to different datasets. For instance, relations in Freebase like *place_of_birth* give much more information about head and tail entities compared to relations like *_similar_to*, *_hypernym* in WordNet. Thus, learning these interactions while training can enable the model to adapt to different datasets.

We address this issue in this paper and propose a novel adaptive framework that allows learning these interactions directly from the data during training. The proposed framework is capable of weighing entities and relations differently by using a fully connected interaction layer. It allows the proposed method to adapt to different datasets by learning dataset-specific interactions. By extensive experiments on multiple benchmark datasets, we show the effectiveness of the proposed method on the link prediction task. We also demonstrate that the proposed method assigns different weights to entities and relations by learning dataset-specific interactions.

In summary, we make the following contributions:

- We propose an adaptive interaction framework that can discover relevant interactions of embeddings from data. We show that many of the existing methods can be seen as special cases of the proposed framework.
- Based on this framework, we propose two new models FCE and FCConvE which outperform the baseline models on link prediction task across commonly used benchmark datasets.
- We also present a method to analyse the fully connected interactions and use it to compare the interactions learned by FCConvE for different datasets.

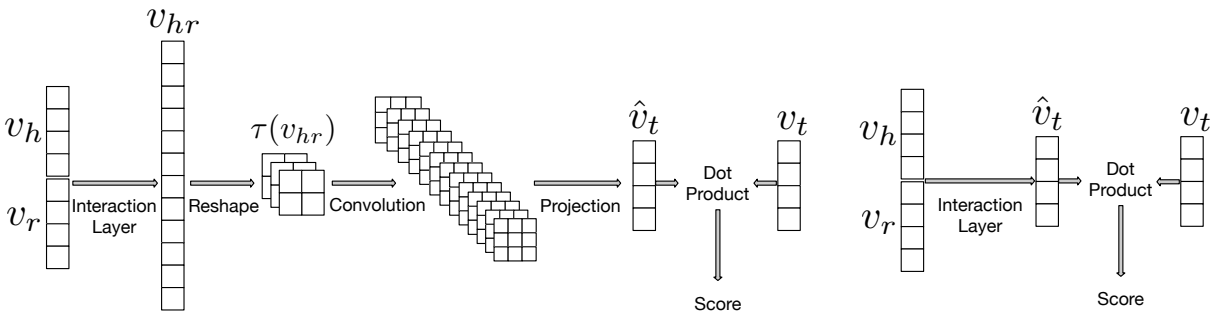
Notations: A Knowledge Graph is represented by $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ where $\mathcal{E}$ is the set of entities, $\mathcal{R}$ is the set of relations and $\mathcal{T} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is the set of triples stored in the graph. Most of the KG embedding methods learn vectors $v_e \in \mathbb{R}^{d_e}$ for $e \in \mathcal{E}$, and $v_r \in \mathbb{R}^{d_r}$ for $r \in \mathcal{R}$. Some methods also learn projection matrices $M_r \in \mathbb{R}^{d_r \times d_e}$ for relations. The correctness of a triple is evaluated using a model specific score function $score : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \rightarrow \mathbb{R}$. For learning the embeddings, a loss function $\mathcal{L}(\mathcal{T}, \mathcal{T}'; \theta)$, defined over a set of positive triples $\mathcal{T}$, a set of (sampled) negative triples $\mathcal{T}'$, and the parameters θ is optimized. I_d denotes the $d \times d$ identity matrix while 0_d denotes the $d \times d$ zero matrix. $diag(v)$ denotes a diagonal matrix created from vector v. All vectors are assumed to be column vectors including the concatenation of vectors $[v_1; v_2; \ldots v_k]$. In case of matrix, $[M_1; M_2]$ denotes the block matrix consisting of blocks M_1 and M_2.

2 Related Work

The problem of learning KG Embeddings for link prediction has been very popular in the last few years. Based on the score function, these methods can be broadly grouped into three categories, namely Additive, Multiplicative and Neural models.

2.1 Additive Models

This is the class of methods where the vectors interact via additive operations after an optional projection operation. One of the simple and popular additive models is TransE (Bordes et al., 2013) where the entity and relation vectors lie in the same vector space. The relation vector acts as a translation from the head entity vector to the tail entity vector.



(a) FCConvE (b) FCE

Figure 3: Architecture diagrams for FCConvE (left) and FCE (right). Please refer to Section 3.3 and Section 3.4 for more details.

SE (Bordes et al., 2011) is another model that uses relation specific similarity between head and tail entity vectors. Following the ideas of translation and projection, many different methods have been developed. These include TransH (Wang et al., 2014), TransR (Lin et al., 2015), STransE (Nguyen et al., 2016), ITransF (Xie et al., 2017), etc. These methods can only extract a restricted set of interactions from the vectors.

2.2 Multiplicative Models

In this class of methods, the vectors interact via a multiplicative operation. One of the initial models in this category is RESCAL (Nickel et al., 2011), which is based on tensor factorization. It models entities as vectors while relations as matrices. DistMult (Yang et al., 2014) is a special case of RESCAL where the relations matrices are restricted to be diagonal. However, DistMult score function is symmetric and hence it can not handle asymmetric relations. To alleviate this issue, HolE (Nickel et al., 2016) was proposed which uses circular correlation operation between head and tail entity vectors. ComplEx (Trouillon et al., 2016) addresses the same issue by modelling vectors in the complex domain. The asymmetry of complex dot product allows it to handle symmetric, asymmetric as well as anti-symmetric relations with the same score function. SimplE (Kazemi and Poole, 2018) is a more recent model based on tensor factorization which can express all types of relations. RotatE (Sun et al., 2019) is another model which uses complex vectors for representation. It models relation vectors as rotations from head to tail entity vector and then uses L1-norm based distance as score function. Similar to additive models, multiplicative models are also restricted in terms of the interactions they can extract.

2.3 Neural Models

There are many models that use various neural network architectures for learning KG Embeddings. Some of these models are NTN (Socher et al., 2013), ER-MLP (Dong et al., 2014), CONV (Toutanova et al., 2015), ProjE (Shi and Weninger, 2017), R-GCN (Schlichtkrull et al., 2017), ConvE (Dettmers et al., 2018), R-MLP-2n (Ravishankar et al., 2017), KG-BERT (Yao et al., 2019), InteractE (Vashishth et al., 2020), etc. Unlike other methods, KG-BERT uses word embeddings for encoding entities and relations. Therefore, the interactions of entity and relation vectors are not directly clear. Among the rest of the models, ProjE can extract interactions only from entity or relation vectors but not both. ConvE can extract interactions from both, entity as well as relation vectors, but they are extracted using predefined permutations. InteractE exploits more sophisticated interactions using feature permutation, checkered reshaping, and circular convolution resulting in improved performance. However, these methods depend on a fixed set of interactions defined by the model and can not adaptively learn these interactions during training.

As described in the next section, the models proposed in this paper are neural models that can adaptively learn the interactions from the entity as well well as relation vectors using a fully connected interaction layer.

3 Proposed Method

3.1 The Adaptive Interaction Framework

As shown in Figure 2, it consists of the following three components

Type	Model	Score Function $\sigma(h,r,t)$	Interactions
Additive	SE (Bordes et al., 2011)	$-\left\|M_r^1 v_h - M_r^2 v_t\right\|_p$	Manual
	TransE (Bordes et al., 2013)	$-\left\|v_h + v_r - v_t\right\|_p$	Manual
	TransR (Lin et al., 2015)	$-\left\|M_r v_h + v_r - M_r v_t\right\|_p$	Manual
	STransE (Nguyen et al., 2016)	$-\left\|M_r^1 v_h + v_r - M_r^2 v_t\right\|_p$	Manual
Multiplicative	DistMult (Yang et al., 2014)	$\langle v_r, (v_h \odot v_t)\rangle$	Manual
	HolE (Nickel et al., 2016)	$\langle v_r, (v_h \star v_t)\rangle$	Manual
	ComplEx (Trouillon et al., 2016)	$\text{Real}(\langle v_r, (v_h \odot \bar{v_t})\rangle)$	Manual
	RotatE (Sun et al., 2019)	$-\left\|v_h \odot v_r - v_t\right\|_p$	Manual
Neural	ER-MLP (Dong et al., 2014)	$\sigma(\langle \beta, \sigma(A \times [v_h; v_r; v_t])\rangle)$	Automatic
	ConvE (Dettmers et al., 2018)	$\langle \sigma(\text{vec}(\sigma(\tau(v_{hr}) * \Omega))U), v_t\rangle$	Manual
Adaptive Interactions Models	FCE (This paper)	$\langle \sigma(W \times [v_h; v_r]), v_t\rangle$	Automatic
	FCConvE (This paper)	$\langle \sigma(\text{vec}(\sigma(\tau(\sigma(W \times [v_h; v_r])) * \Omega))U), v_t\rangle$	Automatic

Table 1: Summary of various Knowledge Graph (KG) embedding methods mentioned in the paper. Here v_h, v_r, v_t denote the head entity, relation and tail entity vectors respectively. M_r, M_r^1, M_r^2 represent the relation specific projection matrices. $\|\cdot\|_p$ denotes the L1-norm ($p = 1$) or L2-norm ($p = 2$). $\langle\cdot,\cdot\rangle$, $\odot$, $\star$ and $*$ represent the inner product, the Hadamard product, circular correlation and convolution operations respectively. A and β represent the first and second layer weight matrices in ER-MLP respectively. Ω represents the convolution filters while U represents the final projection matrix in ConvE. $\tau(v_{hr})$ is a reshaped permutation of $[v_h; v_r]$ with optional repetition. vec denotes vectorization step followed by an activation function σ. Please refer to Section 2 for more details.

3.1.1 Interaction Layer

This layer extracts the interactions between the head entity and relation vectors which are later used for predicting tail entities. The interactions are extracted using a fully-connected (FC) layer:

$$v_{hr} = \sigma(W_r \times [v_h; v_r]), \tag{1}$$

where σ is an activation function. In the general case, W_r is $\mathbb{R}^{d_i \times (d_e + d_r)}$ where d_i is the dimension of the interaction layer. A special case of interaction layer is when $W_r = W, \forall\, r \in \mathcal{R}$ which avoids over-parameterization. We use this special case for our experimentations.

3.1.2 Prediction Layer

This layer predicts a vector $\hat{v}_t$ for the candidate tail entity as

$$\hat{v}_t = g(v_{hr}), \tag{2}$$

where the function g depends on the method. Many of the methods like TransE and DistMult use identity function. For our experiments, we use ConvE's (Dettmers et al., 2018) architecture for this step.

3.1.3 Matching Layer

This is the final layer where the predicted tail entity is matched against a given tail entity producing the final score for the triple. The score function is given as

$$score(h, r, t) = f(\hat{v}_t, v_t). \tag{3}$$

Again, the matching function f is method dependent. We use the vector dot product for matching.

3.2 Existing models as special case

In this section, we demonstrate how the proposed framework generalizes many of the existing models. The score functions for these models can be found in Table 1.

TransE: In TransE, the entities and relations lie in the same $d_e = d_r = d$ dimensional space $\mathbb{R}^d$. The interaction matrix is $W_r = [I_d; I_d]$ with identity activation while the tail prediction function $\hat{v}_t = g(v_{hr}) = v_{hr}$ is the identity function. The matching function takes the form of a vector norm $-\left\|\hat{v}_t - v_t\right\|_p$.

STransE: STransE (Nguyen et al., 2016) generalizes many translation-based models like TransH (Wang et al., 2014) and TransR (Lin et al., 2015) by using two relation specific projection matrices, M_r^1 for head and M_r^2 for tail entities. The interaction matrix is $W_r = [M_r^1; I_{d_r}]$ while the activation and prediction functions are identity functions. The matching function takes the form of $-\left\|\hat{v}_t - M_r^2 v_t\right\|_p$.

DistMult: For DistMult (Yang et al., 2014), the interaction matrix is $W_r = [diag(v_r), 0_d]$ while the activation and prediction functions are identity functions. Vector dot product between $\hat{v}_t$ and v_t is used as matching function.

ConvE: ConvE (Dettmers et al., 2018) is a recent

	FB15K	FB15K-237	WN18	WN18RR
#Entities	14,941	14,541	40,943	40,943
#Relations	1,345	237	18	11
#Train	483,142	272,115	141,441	86,835
#Validation	50,000	17,535	5,000	3,034
#Test	59,071	20,466	5,000	3,134

Table 2: Details of the datasets used in our experiments

model which uses a convolution network for extracting features from the interactions and then predicting the tail entity. Here, the interaction matrix is a fixed and manually specified permutation matrix, used for dimension shuffling in ConvE. Please note that the interaction matrix is shared among all relations. The prediction function is the 2D convolution network applied on the interactions vector v_{hr} while dot product between $\hat{v}_t$ and v_t is used for matching.

Based on the proposed framework, we present two new models FCConvE and FCE in the following sections. Their architecture can be found in Figure 3.

3.3 FCConvE

One issue with the existing methods is that the interactions are either fixed or it requires to be specified manually. For example, in the case of ConvE the permutations are specified by the user. This leaves the choice of interactions to the user and also does not specify which ones are better than others. Also, the performance on the link prediction task varies with different choices of permutations in ConvE. Hence it will be useful if the optimal interactions can be learned directly from the dataset.

FCConvE addresses this exact issue and allows the model to learn the appropriate interactions while training. It achieves this by using interaction matrix $W_r = W, \forall r \in \mathcal{R}$ as model parameter. Since many of the existing interactions can be achieved using different W matrices, the proposed model is capable of choosing the optimal interactions by itself. The score function for FCConvE can be given as follows:

$$score(h, r, t) = \langle \sigma(\text{vec}(\sigma(\tau(v_{hr}) * \Omega))U), v_t \rangle, \tag{4}$$

where v_{hr} is given by (1). As compared to ConvE, FCConvE contains approximately $\mathcal{O}(d_i \times (2d_e + d_r))$ more parameters due to the interaction and projection layers weights.

3.4 FCE

We experiment with a modification of DistMult named DistMult-BCE which uses Binary Cross Entropy (BCE) loss instead of margin-based ranking loss used in (Yang et al., 2014). We also introduce its adaptive interactions variant FCE (Fully Connected Embedding) which uses an FC layer as in (1) for interactions instead of Hadamard product between v_h and v_r. The score function for FCE can be given as follows:

$$score(h, r, t) = \langle \sigma(W \times [v_h; v_r]), v_t \rangle. \tag{5}$$

We use ReLU for the activation function σ. Please note that, similar to DistMult, entity and relation vectors lie in same $d_e = d_r = d$ dimensional space $\mathbb{R}^d$. The interaction vector v_{hr} also lies in the same space $\mathbb{R}^d$ (i.e. $d_i = d$). Unlike DistMult, the interaction matrix $W \in \mathbb{R}^{d \times d}$ is shared across relations. In terms of model size, FCE contains $d_i \times (d_e + d_r) = 2d^2$ more parameters as compared to DistMult.

3.5 Analysing Interactions using NAIW

In this section, we introduce a novel method to analyse fully connected interactions. As mentioned in previous sections, the interaction weight matrix W in FCConvE as well as FCE is shared among all relations i.e. $W_r = W, \forall r \in \mathcal{R}$. This interaction weight matrix can be split into two parts, $W = [W^{\mathcal{E}}; W^{\mathcal{R}}], W^{\mathcal{E}} \in \mathbb{R}^{d_i \times d_e}, W^{\mathcal{R}} \in \mathbb{R}^{d_i \times d_r}$ corresponding to the head entity and the relation vectors respectively. The equation for the interaction layer can be re-written as follows:

$$v_{hr} = \sigma(W^{\mathcal{E}} v_h + W^{\mathcal{R}} v_r). \tag{6}$$

The values in $W^{\mathcal{E}}$ and $W^{\mathcal{R}}$ represent the importance of various dimensions of the head entity and relation vectors. We use the absolute values of these weights for comparing the importance of entity and relation. Specifically, we use the *Normalized Absolute Interaction Weights (NAIW)* as defined below for comparing the weights corresponding to entities and relations.

$$\begin{aligned} V^{\mathcal{E}} &= \sum_{j=1}^{d_e} \text{AbsoluteValue}(W^{\mathcal{E}}[:, j]), \\ \text{NAIW}^{\mathcal{E}} &= \frac{V^{\mathcal{E}}}{\max(V^{\mathcal{E}})}, \end{aligned} \tag{7}$$

where $V^{\mathcal{E}} \in \mathbb{R}^{d_i}$ is a vector containing sum of absolute interaction weights across dimensions of

Model	FB15K-237					WN18RR				
	MR↓	MRR↑	Hits↑			MR↓	MRR↑	Hits↑		
			@1	@3	@10			@1	@3	@10
DistMult (Yang et al., 2014)	254	24.1	15.5	26.3	41.9	5110	43.0	39.0	44.0	49.0
ComplEx (Trouillon et al., 2016)	339	24.7	15.8	27.5	42.8	5261	44.0	41.0	46.0	51.0
R-GCN (Schlichtkrull et al., 2017)	-	24.8	15.3	25.8	41.7	-	-	-	-	-
ConvE (Dettmers et al., 2018)	**244**	32.5	23.7	35.6	50.1	4187	43.0	40.0	44.0	52.0
DistMult-BCE	318	30.1	21.6	33.0	46.9	7037	41.0	38.6	41.6	46.0
FCE	331	30.6	21.7	33.7	48.6	4732	41.3	38.2	42.2	47.5
FCConvE	255	**35.5**	**26.4**	**39.1**	**54.0**	**4103**	**46.1**	**42.8**	**47.7**	**52.7**

Model	FB15K					WN18				
	MR↓	MRR↑	Hits↑			MR↓	MRR↑	Hits↑		
			@1	@3	@10			@1	@3	@10
TransE (Bordes et al., 2013)	-	46.3	29.7	57.8	74.9	-	49.5	11.3	88.8	94.3
DistMult (Yang et al., 2014)	97	65.4	54.6	73.3	82.4	902	82.2	72.8	91.4	93.6
ComplEx (Trouillon et al., 2016)	-	69.2	59.9	75.9	84.0	-	94.1	93.6	93.6	94.7
R-GCN (Schlichtkrull et al., 2017)	-	69.6	60.1	76.0	84.2	-	81.4	69.7	92.9	**96.4**
ConvE (Dettmers et al., 2018)	**51**	65.7	55.8	72.3	83.1	**374**	94.3	93.5	94.6	95.6
DistMult-BCE	115	73.3	66.8	77.8	84.9	671	83.9	74.8	92.6	94.7
FCE	108	**74.6**	**67.8**	**79.5**	**86.1**	516	94.2	93.6	94.5	95.2
FCConvE	67	71.7	63.4	77.3	85.6	440	**94.8**	**94.3**	**95.1**	95.5

Table 3: Link prediction results on benchmark datasets. Here ↑ indicates higher values are better while ↓ indicates lower values are better. The adaptive interaction versions of the models FCConvE and FCE outperform the corresponding baseline models ConvE and DistMult-BCE in all the datasets. They also outperform other methods across all datasets. Results for the baseline models except TransE were taken from (Dettmers et al., 2018). For TransE, we have taken the results from (Nickel et al., 2016). We have also included results for a modification of DistMult called DistMult-BCE. Please refer to Section 4.2 for more details.

head entity. It denotes the importance of the head entity for each unit in the interaction layer. We normalize this vector such that the values lie in $[0, 1]$ range. This allows us to compare this value across datasets. Similarly, we can calculate $V^{\mathcal{R}}$ and NAIW$^{\mathcal{R}}$ which denote the importance of relation for units in the interaction layer.

Each value in the NAIW vector represents the importance of entities (for NAIW$^{\mathcal{E}}$) or relations (for NAIW$^{\mathcal{R}}$) for the link prediction task. Thus, comparing these values helps us understand the relative importance of entities and relations for different datasets. Since a comparison of individual interaction units may be inconclusive, we compare their distributions. We estimate the distributions[1] of NAIW$^{\mathcal{E}}$ and NAIW$^{\mathcal{R}}$ and compare them across multiple datasets. These distributions allow us to compare the importance of the entity and relation for the link prediction task.

[1]We use *gaussian_kde* function from SciPy library for estimating distributions.

4 Experiment Results

In this section, we evaluate the proposed method on the link prediction task and compare it against the baselines. The details of the datasets used for evaluation are given in Table 2. We provide the implementation details followed by the results in the following sections.

4.1 Implementation details

In our experiments, we use 200-dimension embeddings for both entity as well as relations. For selecting other hyper-parameters, we use cross-validation using the MRR on validation split of the data. Similar to ConvE, we use dropouts and batch normalization at input, convolution and final projection layer. The corresponding dropout probabilities are selected from [0.1, 0.2, 0.3], [0.2, 0.3, 0.4] and [0.4, 0.5, 0.6] respectively. For the interaction layer, we use 5000 dimensions reshaped to $25 \times 10 \times 20$ before applying convolution. Unlike ConvE, FCConvE uses depthwise group convolution. The

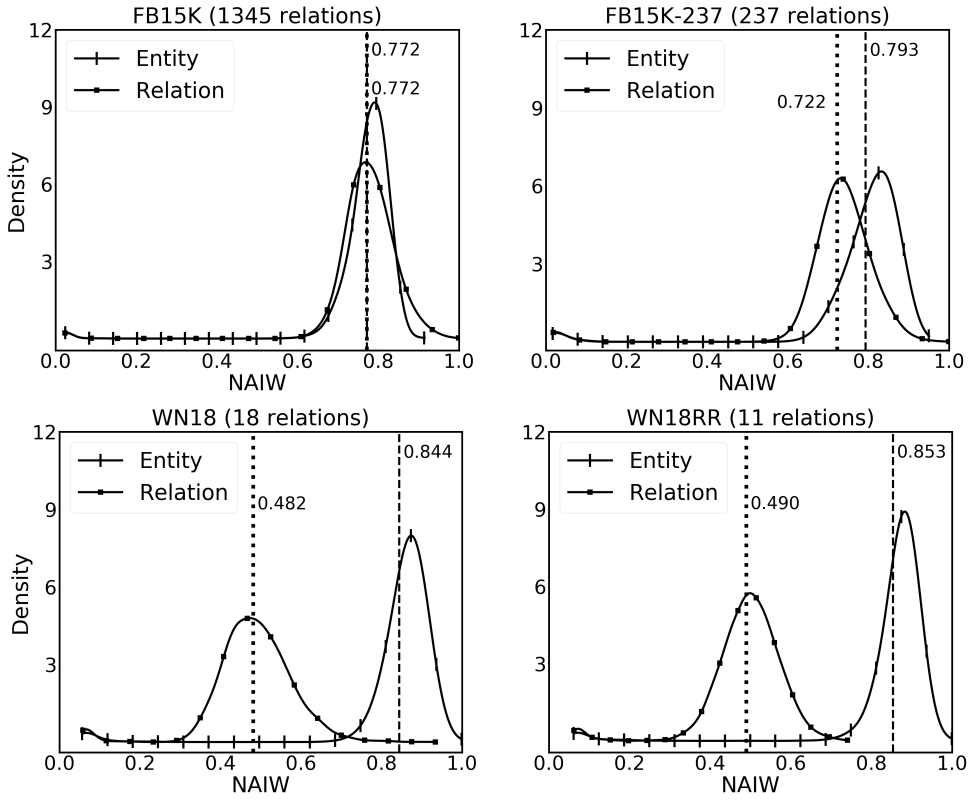


Figure 4: Distributions of the Normalized Absolute Interaction Weights for entities and relations learned by FCConvE on different datasets. The means of these distributions are shown as a dashed (for Entity) or dotted (for Relation) vertical lines along with their values. The datasets are arranged according to decreasing order of number of relations from left to right. As we can see, for lower number of relations, the difference in weights for entities and relations are much higher. Please refer to Section 4.3 for more details.

filter size for convolution is selected from [3×3, 5×5]. For optimization, we use Adam optimizer with an initial learning rate of 0.001. We use early stopping using MRR on validation split with the maximum number of epochs set to 100. The best model is then run for 1000 epochs[2] and final performance is reported. We use 500 negative samples per correct triple. For head entity prediction, we follow ConvE and use reversed relations during training as well as evaluation. The embeddings are trained using binary cross-entropy loss.

4.2 Link Prediction

Given a test or validation triple, we score the head entity and relation against all entities and report the rank of the correct tail entity. A similar strategy is used for head entity prediction except that we use reverse relation vectors. The model's performance is evaluated on both head as well as tail entity prediction and the average performance is reported. Similar to previous work, we use filtered setting, i.e., we exclude all triples appearing in train, test or validation split while ranking. We report Mean Reciprocal Rank (MRR), Mean Rank (MR) and Hits@k for k=1, 3, 10 on test split. The results can be found in Table 3.

For comparison, we use a few representative baselines from each category of the models. Specifically, we use TransE for additive models, DistMult and ComplEx for multiplicative models, and R-GCN and ConvE for neural models. Please note that our goal is to compare a model with its adaptive interaction version, instead of comparing all available models.

We observe from the results that the proposed adaptive interaction versions of the models outperform the corresponding baseline models in all the datasets. FCConvE significantly outperforms ConvE in all the datasets except WN18 where the performance is comparable. Similarly, FCE significantly outperforms DistMult-BCE in WN18 while showing marginal improvements is other datasets. Also, they outperform other methods on the link prediction task across all the datasets suggesting

[2]We pick the model from the epoch with the best validation split MRR, and it need not be the final epoch.

that the proposed approch is able to adapt to different datasets resulting in performance improvements.

It should be noted that ConvE would struggle to differentially weigh entity and relation vector dimensions due to the sharing of convolution filters. Adding an FC interaction layer allows it to prioritize between entity and relation vectors resulting in better performance of FCConvE. We also observe that the DistMult-BCE model significantly outperforms the DistMult model in FB15K and FB15K-237 which suggests the BCE loss with multiple negative samples improves performance.

Among the metrics used in Table 3, MR is more sensitive to outliers (i.e., large values of ranks) than others. We observe that the proposed approaches achieve improvements in MRR and Hits@k, but not MR for many datasets. It suggests that the proposed methods are effective in bringing more cases into the high-rank region, which could be a desirable property in many applications.

4.3 Interactions Analysis

In this section, we analyse the interactions learned by FCConvE on different datasets. We use the distributions of NAIW$^{\mathcal{E}}$ and NAIW$^{\mathcal{R}}$, as defined in Section 3.5 and compare them. The results are shown in Figure 4.

As we can see from Figure 4, the distributions of NAIW$^{\mathcal{E}}$ and NAIW$^{\mathcal{R}}$ varies across different datasets. Furthermore, we make the following observations.

- The difference between the means of NAIW$^{\mathcal{E}}$ and NAIW$^{\mathcal{R}}$ increases with decreasing number of relations. Among the datasets used, FB15K has the most number of relations (i.e. 1,345) while WN18RR has the least number of relations (i.e. 11). This number is correlated with the difference of the means of NAIW$^{\mathcal{E}}$ and NAIW$^{\mathcal{R}}$ with WN18RR having the highest difference, while FB15K having the lowest difference. This suggests that when a dataset has a small number of relations, entities have more distinguishing capability than the relations.

- The relations in Freebase datasets (i.e. FB15K and FB15K-237) have more distinguishing capability than relations in WordNet datasets (i.e. WN18 and WN18RR). For example, relations like *place_of_birth* in Freebase restricts candidate entity types for head and tail entities. On the other hand, relations in Wordnet (e.g. *_similar_to*, *_hypernym*) are not very specific to some type of entities. This behavior is reflected in the distributions of NAIW$^{\mathcal{E}}$ and NAIW$^{\mathcal{R}}$ with relations getting more weights in Freebase datasets as compared to WordNet datasets.

4.4 Effect of various interactions on ConvE

To further understand the advantages of adaptive interactions, we compare its performance with various fixed interactions (as used in (Vashishth et al., 2020)). As mentioned in Section 3.2, ConvE can use a permutation matrix for generating interactions. For demonstration, let's assume a head entity vector $v_h = [v_h^1, v_h^2, v_h^3, v_h^4, v_h^5, v_h^6]$ and a relation vector $v_r = [v_r^1, v_r^2, v_r^3, v_r^4, v_r^5, v_r^6]$. These vectors are concatenated, permuted and then reshaped into a 4×3 matrix before passing it to the convolution layer. As shown in Figure 5, the following are some of the candidate permutations.

Plain: A plain concatenation and reshaping of the vectors.

Alternate Rows: Alternate rows of head entity and relation vectors dimensions.

Alternate: Strictly alternating dimensions of head entity and relation vectors.

$$\begin{bmatrix} v_h^1 & v_h^2 & v_h^3 \\ v_h^4 & v_h^5 & v_h^6 \\ v_r^1 & v_r^2 & v_r^3 \\ v_r^4 & v_r^5 & v_r^6 \end{bmatrix} \quad \begin{bmatrix} v_h^1 & v_h^2 & v_h^3 \\ v_r^1 & v_r^2 & v_r^3 \\ v_h^4 & v_h^5 & v_h^6 \\ v_r^4 & v_r^5 & v_r^6 \end{bmatrix} \quad \begin{bmatrix} v_h^1 & v_r^1 & v_h^2 \\ v_r^2 & v_h^3 & v_r^3 \\ v_h^4 & v_r^4 & v_h^5 \\ v_r^5 & v_h^6 & v_r^6 \end{bmatrix}$$

(a) Plain (b) Alternate Rows (c) Alternate

Figure 5: Some example permutations of two 6-dimensional vectors v_h and v_r.

As we can see, *Plain* method allows entity-relation interactions only at the boundary region while *Alternate Rows* and *Alternate* allow deeper interactions.

We run the best hyper-parameters settings of ConvE with these three permutations on all the datasets and compare the MRR of the link prediction task. As seen from the results in Table 4, the *Alternate Rows* and *Alternate* permutation schemes achieve better results compared to *Plain* permutation scheme. However, since FCConvE can learn

Permutation	FB15K	FB15K-237	WN18	WN18RR
Plain	63.2	32.7	94.3	43.2
Alternate Rows	63.6	33.3	94.8	44.3
Alternate	63.9	33.3	94.8	44.4
FCConvE	71.7	35.5	94.8	46.1

Table 4: The effect of various permutation schemes on the performance of ConvE. We report the MRR in link prediction task across various datasets. As we can see, the performance of ConvE is dependent on the choice of permutation scheme and using Alternate or Alternate Rows permutation improves the performance of ConvE. Please refer to Section 4.4 for more details.

the interactions while training, it achieves better or comparable MRR on all the datasets.

5 Conclusions and Future Work

We presented an adaptive interaction framework for learning KG Embeddings and proposed two new models based on the framework. We demonstrated that the proposed models are capable of learning relevant interactions across different datasets. We also demonstrated how some of the existing KG Embedding models can be seen as special cases of the proposed framework.

In the future, we would like to further analyze the interaction layer and its correlation with more dataset properties.

Acknowledgments

We thank the anonymous reviewers for their constructive comments. This work is supported by the Ministry of Human Resources Development (Government of India).

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, pages 1811–1818.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM.

Zhipeng Huang, Bogdan Cautis, Reynold Cheng, and Yudian Zheng. 2016. Kb-enabled query recommendation for long-tail queries. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 2107–2112, New York, NY, USA. ACM.

Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. In *Advances in Neural Information Processing Systems*, pages 4284–4295.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-ending learning. In *Proceedings of AAAI*.

Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. 2016. STransE: a novel embedding model of entities and relationships in knowledge bases. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 460–466, San Diego, California. Association for Computational Linguistics.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. 2016. Holographic embeddings of knowledge graphs. In *AAAI*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*, volume 11, pages 809–816.

Srinivas Ravishankar, Chandrahas, and Partha Pratim Talukdar. 2017. Revisiting simple neural networks for learning representations of knowledge graphs. *6th Workshop on Automated Knowledge Base Construction (AKBC) at NIPS 2017*.

M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling. 2017. Modeling Relational Data with Graph Convolutional Networks. *ArXiv e-prints*.

Sanket Shah, Anand Mishra, Naganand Yadati, and Partha Pratim Talukdar. 2019. Kvqa: Knowledge-aware visual question answering. In *AAAI*.

Baoxu Shi and Tim Weninger. 2017. Proje: Embedding projection for knowledge graph completion. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *WWW*.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Lisbon, Portugal. Association for Computational Linguistics.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, Nilesh Agrawal, and Partha Talukdar. 2020. Interacte: Improving convolution-based knowledge graph embeddings by increasing feature interactions. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, pages 3009–3016. AAAI Press.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119. Citeseer.

Qizhe Xie, Xuezhe Ma, Zihang Dai, and Eduard Hovy. 2017. An interpretable knowledge transfer model for knowledge base completion. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 950–962, Vancouver, Canada. Association for Computational Linguistics.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Kg-bert: Bert for knowledge graph completion. *ArXiv*, abs/1909.03193.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with Freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 956–966, Baltimore, Maryland. Association for Computational Linguistics.

Inducing Interpretability in Knowledge Graph Embeddings

Chandrahas[1] **Tathagata Sengupta**[2†*] **Cibi Pragadeesh**[3*†] **Partha Talukdar**[1]
[1]Indian Institute of Science, Bangalore, [2]Adobe, Bangalore,
[3]University of California, Los Angeles
chandrahas@iisc.ac.in, tathagatasengupta3@gmail.com
cibi.pragadeesh@gmail.com, ppt@iisc.ac.in

Abstract

We study the problem of inducing interpretability in Knowledge Graph (KG) embeddings. Learning KG embeddings has been an active area of research in the past few years, resulting in many different models. However, most of these methods do not address the interpretability (semantics) of individual dimensions of the learned embeddings. In this work, we study this problem and propose a method for inducing interpretability in KG embeddings using entity co-occurrence statistics. The proposed method significantly improves the interpretability, while maintaining comparable performance in other KG tasks.

1 Introduction

Knowledge Graphs such as Freebase (Bollacker et al., 2008) and NELL (Mitchell et al., 2015) have become important resources for supporting many AI applications like web search, Q&A, etc. They store a collection of facts in the form of a graph. The nodes in the graph represent real world entities such as *Roger Federer*, *Tennis*, *United States* etc while the edges represent relationships between them.

These KGs have grown huge, but they are still not complete (Toutanova et al., 2015). Hence the task of inferring new facts becomes important. KG embeddings have been a popular approach for this task as they can perform the inference task efficiently. This task has achieved significant attention in the literature and many methods have been proposed, such as, (Bordes et al., 2013; Riedel et al., 2013; Yang et al., 2014; Toutanova et al., 2015; Trouillon et al., 2016; Schlichtkrull et al., 2017; Dettmers et al., 2018; Balazevic et al., 2019), etc. These methods learn representations for entities and relations as vectors in a vector space, capturing global information about the KG. The task of KG inference is then defined as operations over these vectors. Some of these methods like (Riedel et al., 2013) and (Toutanova et al., 2015) are capable of exploiting additional text data apart from the KG, resulting in better representations.

Although these methods have shown good performance in the end task, they do not address the interpretability, i.e., understanding semantics of individual dimensions of the KG embedding. Such representations enable a better understanding of the model and can be helpful for explaining a model's decision on an end application.

In this work, we focus on incorporating interpretability in KG embeddings. Specifically, we aim to learn interpretable embeddings for KG entities by incorporating additional entity co-occurrence statistics from text data. This work is motivated by (Lau et al., 2014) who presented automated methods for evaluating topics learned via topic modelling methods. We adapt these methods for KG embedding models and propose a method to directly maximize them while learning KG embedding. As demonstrated by the experiments, we find that such modeling significantly improves interpretability, supporting our choice of using topic coherence for embedding dimensions. To the best of our knowledge, this work presents the first regularization term which induces interpretability in KG embeddings.

2 Related Work

Several methods have been proposed for learning KG embeddings. They differ on the modeling of entities and relations, usage of text data and interpretability of the learned embeddings. We summarize some of these methods in following sections.

[*]Contributed equally to the work.
[†]This research was conducted during the authors' internships at Indian Institute of Science, Bangalore.

Proceedings of the 17th International Conference on Natural Language Processing, pages 70–75
Patna, India, December 18 - 21, 2020.

2.1 KG Embedding models

Most of the KG embedding models represent entities and relations as vectors in $\mathbb{R}^{d_e}$ and $\mathbb{R}^{d_r}$ respectively (usually, d_e=d_r). A score function uses these vectors to calculate the correctness of a given triple. Based on the score function, these methods can be categorized as additive models (Bordes et al., 2013; Lin et al., 2015; Xiao et al., 2015; Xie et al., 2017), multiplicative models (Nickel et al., 2011; Yang et al., 2014; Trouillon et al., 2016; Balazevic et al., 2019) and nueral models (Dong et al., 2014; Dettmers et al., 2018). There are other methods which are able to incorporate text data while learning KG embeddings. For example, the method proposed in (Riedel et al., 2013) assumes a combined universal schema of relations from KG as well as text. This method is further improved in (Toutanova et al., 2015) using textual relation encoder allowing parameter sharing among similar textual relations. However, none of these methods address the interpretability of the embeddings.

2.2 Interpretability of Embeddings

While the KG embedding models perform well in many tasks, the semantics of learned representations are not directly clear. This problem for word embeddings has been addressed in (Murphy et al., 2012; Faruqui et al., 2015; Subramanian et al., 2018) where they apply a set of constraints inducing interpretability. A similar task of learning semantic features for entities and relations is KG was addressed in (Xiao et al., 2016). However, their approach is not applicable for the much popular KG embedding methods. The model proposed in (Xie et al., 2017) can generate interpretable embeddings for relations, but not entities. Another approach, as proposed in (Gusmao et al., 2018), is to generate weighted Horn rules as explanations for link prediction. We refer the reader to Section 4 of (Bianchi et al., 2020) for further reading in this direction.

Our method differs from the previous works in the following aspects. Firstly, we focus on learning interpretable embeddings for KG entities rather than relations. Second, we incorporate side information about entities instead of constraints for inducing interpretability. Third, we use vector space modeling rather than probabilistic modelling (as in (Xiao et al., 2016)) allowing the proposed method to be applicable to many existing KG embedding models.

3 Proposed Method

The proposed method is motivated by a measure of coherence in topic modelling literature (Lau et al., 2014). This measure allows an automated evaluation of the quality of topics learned by topic modeling methods by using additional Point-wise Mutual Information (PMI) for word pairs. It was also shown to have high correlation with human evaluation of topics.

Based on this measure of coherence, we propose a regularization term. This term can be used with existing KG embedding methods for inducing interpretability. It is described in the following sections.

3.1 Coherence

In topic models, coherence of a topic can be determined by semantic relatedness among top entities within the topic. This idea can also be used in vector space models by treating dimensions of the vector space as topics. With this assumption, we can use a measure of coherence defined in following section for evaluating interpretability of the embeddings.

3.1.1 $Coherence@k$

Coherence for top k entities along dimension l is defined as follows.

$$Coherence@k^{(l)} = \sum_{i=2}^{k} \sum_{j=1}^{i-1} p_{ij} \qquad (1)$$

where p_{ij} is PMI score between entities e_i and e_j extracted from text data. It is given as follows

$$p_{ij} = \log \left(\frac{Pr(e_i, e_j)}{Pr(e_i) \times Pr(e_j)} \right). \qquad (2)$$

Here, $Pr(e_i, e_j)$ represents the joint probability of co-occurrence of entities e_i and e_j, while $Pr(e_i)$ and $Pr(e_j)$ represent the corresponding marginal probabilities, pre-computed using an auxiliary corpus.

$Coherence@k$ has been shown to have high correlation with human interpretability of topics learned via various topic modeling methods(Lau et al., 2014). Hence, we can expect interpretable embeddings by maximizing it.

$Coherence@k$ for the entity embedding matrix θ_e is defined as the average over all dimensions.

$$Coherence@k = \frac{1}{d} \sum_{l=1}^{d} Coherence@k^{(l)}. \qquad (3)$$

3.1.2 Inducing coherence while learning embeddings

We want to learn an embedding matrix θ_e which has high coherence (i.e., which maximizes $Coherence@k$). Since θ_e changes during training, the set of top k entities along each dimension varies over iterations. Hence, directly maximizing $Coherence@k$ may not be feasible.

An alternative approach could be to promote higher values for entity pairs having high PMI score p_{ij}. This will result in an embedding matrix θ_e with a high value of $Coherence@k$ since high PMI entity pairs are more likely to be among top k entities.

This idea can be captured by following coherence term

$$\mathcal{C}(\theta_e, P) = \sum_{i=2}^{n} \sum_{j=1}^{i-1} \|v(e_i)^\intercal v(e_j) - p_{ij}\|^2 \quad (4)$$

where P is entity-pair PMI matrix and $v(e)$ denote vector for entity e. This term can be used in the objective function defined in (7).

3.2 Entity Model (Model-E)

We use the Entity Model proposed in (Riedel et al., 2013) for learning KG embeddings. However, it should be noted that the proposed regularizer can be used along with any KG embedding model which represents entities as vectors. Also, as pointed in (Kadlec et al., 2017; Ruffinelli et al., 2020; Jain et al., 2020), various KG embedding models achieve similar performances when trained properly. Therefore, we select Model-E which is simple yet effective. This model assumes a vector $v(e)$ for each entity and two vectors $v_s(r)$ and $v_o(r)$ for each relation of the KG. The score for the triple (e_s, r, e_o) is given by,

$$f(e_s, r, e_o) = v(e_s)^\intercal v_s(r) + v(e_o)^\intercal v_o(r). \quad (5)$$

Training these vectors requires incorrect triples. So, we use the closed world assumption. For each triple $t \in \mathcal{T}$, we create two negative triples t_o^- and t_s^- by corrupting the object and subject of the triples respectively such that the corrupted triples do not appear in training, test or validation data. The loss for a triple pair is defined as $loss(t, t^-) = -\log(\sigma(f(t) - f(t^-)))$. Then, the aggregate loss function is defined as

$$L(\theta_e, \theta_r, \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \left(loss(t, t_o^-) + loss(t, t_s^-) \right). \quad (6)$$

3.3 Objective

The overall loss function can be written as follows

$$L(\theta_e, \theta_r, \mathcal{T}) + \lambda_c \mathcal{C}(\theta_e, P) + \lambda_r \mathcal{R}(\theta_e, \theta_r) \quad (7)$$

where $\mathcal{R}(\theta_e, \theta_r) = \frac{1}{2} \left(\|\theta_e\|^2 + \|\theta_r\|^2 \right)$ is the $L2$ regularization term and λ_c and λ_r are hyper-parameters controlling the trade-off among different terms in the objective function.

4 Experiments and Results

4.1 Datasets

We use the FB15k-237 (Toutanova and Chen, 2015) dataset, a factual KG, for experiments. It contains 14541 entities and 237 relations. The triples are split into training, validation and test set having 272115, 17535 and 20466 triples respectively. For extracting entity co-occurrences, we use the textual relations used in (Toutanova et al., 2015). It contains around 3.7 millions textual triples, which we use for calculating PMI for entity pairs.

4.2 Experimental Setup

We use the method proposed in (Riedel et al., 2013) as the baseline. Please refer to Section 3.2 for more details. For evaluating the learned embeddings, we test them on different tasks. All the hyper-parameters are tuned using performance (MRR) on validation data. We use 100 dimensions after cross validating among 50, 100 and 200 dimensions. For regularization, we use $\lambda_r = 0.01$ (from $10, 1, 0.1, 0.01$) and $\lambda_c = 0.01$ (from $10, 1, 0.1, 0.01$) for $L2$ and coherence regularization respectively. We use multiple random initializations sampled from a Gaussian distribution. For optimization, we use gradient descent and stop optimization when gradient becomes 0 upto 3 decimal places. The final performance measures are reported for test data.

4.3 Results

In following sections, we compare the performance of the proposed method with the baseline method in different tasks. Please refer to Table 1 for results.

Method	Link Prediction		
	MRR↑	MR↓	Hits@10(%)↑
Baseline	$\mathbf{31.6 \pm 0.08}$	121.9 ± 1.80	$\mathbf{48.3 \pm 0.39}$
Proposed	30.4 ± 0.08	$\mathbf{111.9 \pm 1.12}$	46.8 ± 0.08
	Triple Classification		
	AUC(%)↑	Accuracy(%)↑	
Baseline	72.9 ± 0.16	63.2 ± 0.50	
Proposed	$\mathbf{73.2 \pm 0.28}$	$\mathbf{67.6 \pm 0.17}$	
	Interpretability		
	AutoWI@5(%)↑	Coherence@5↑	Manual WI(%)↑
Baseline	6 ± 4.14	-47.4 ± 4.68	12
Proposed	$\mathbf{66 \pm 5.89}$	$\mathbf{-12.5 \pm 4.48}$	**84**

Table 1: Results of various tasks on FB15k-237 dataset. Here ↑ indicates higher values are better while ↓ indicates lower values are better. The proposed method significantly improves interpretability while maintaining comparable performance on KG tasks (4.3).

Top 5
Baseline
-**Jurist**, **Pipe organ**, USA, **Lions Gate Entertainment**, UK
-Guitar, **71st Academy Awards**, **Jurist**, Piano, Bass guitar
-Actor, **Official Website**, Screenwriter, Film Producer, **USA**
-**Jurist**, USA, **Marriage**, **Male**, UK
-**Pipe organ**, **Official Website**, Actor, Film Producer, Screenwriter
Proposed Method
-Juris Doctor, Business Administration, Biology, Psychology, BS
-Bachelor of Arts, PhD, Bachelor's degree, BS, MS
-European Union, Europe, Netherlands, Portugal, **Government**
-UK, Hollywood, **DVD**, London, Europe
-Hollywood, Academy Awards, **USA**, DVD, **Los Angeles**

Table 2: Top 5 entities for randomly selected dimensions. As we see, the proposed method produces more coherent entities compared to the baseline. Incoherent entities are marked in bold face. [1]

4.3.1 Interpretability

For evaluating the interpretability, we use $Coherence@k$ (3), automated and manual word intrusion tests. In word intrusion test (Chang et al., 2009), top $k(=5)$ entities along a dimension are mixed with the bottom most entity (the intruder) in that dimension and shuffled. Then multiple (3 in our case) human annotators are asked to find out the intruder. We use majority voting to finalize one intruder. Amazon Mechanical Turk was used for crowdsourcing the annotation task and we used 25 randomly selected dimensions for evaluation. Thus, each of the three annotators evaluates 25 examples. For automated word intrusion (Lau et al., 2014), we calculate following score for all $k+1$ entities

$$\text{AutoWI}(e_i) = \sum_{j=1, j\neq i}^{k+1} p_{ij} \tag{8}$$

where p_{ij} are the PMI scores. The entity having least score is identified as the intruder. We report the fraction of dimensions for which we were able to identify the intruder correctly.

As we can see in Table 1, the proposed method achieves better values for $Coherence@5$ as a direct consequence of the regularization term, thereby maximizing coherence between appropriate entities. Performance on the word intrusion task also improves drastically as the intruder along each dimension is a lot easier to identify owing to the fact that the top entities for each dimension group together more conspicuously.

4.3.2 Link Prediction

In this experiment, we test the model's ability to predict the best object entity for a given subject entity and relation. For each of the triples, we fix the subject and the relation and rank all entities (within same category as true object entity) based on their score according to (5). We report Mean Rank (MR) and Mean Reciprocal rank (MRR) of the true object entity and Hits@10 (the number of times true object entity is ranked in top 10) as percentage. A good model should have higher values for MRR and Hits@10, and lower value for MR.

The coherence regularization term's objective, being tangential to that of the original loss function, is not expected to affect the link prediction task's performance. However, the results show a trivial drop of 1.2 in MRR. Upon further inspection, we found that the coherence term gives credibility to certain triples otherwise deemed incorrect by the closed world assumption. These triples appear in the text corpus and contain entity pairs with high PMI values.

4.3.3 Triple Classification

In this experiment, we test the model on classifying correct and incorrect triples. For finding incorrect triples, we corrupt the object entity with a randomly selected entity within the same category. For classification, we use validation data to find the best threshold for each relation by training an SVM classifier and later use this threshold for classifying test triples. We report the mean accuracy and mean AUC over all relations.

We observe that the proposed method achieves slightly better performance for triple classification

[1] We have used abbreviations for BS (Bachelor of Science), MS (Master of Science), UK (United Kingdom) and USA (United States of America). They appear as full form in the data.

improving the accuracy by 4.4. The PMI information adds more evidence to the correct triples which are related in text data, generating a better threshold that more accurately distinguishes correct and incorrect triples.

4.4 Qualitative Analysis of Results

Since our aim is to induce interpretability in representations, in this section, we evaluate the embeddings learned by the baseline as well as the proposed method. For both methods, we select some dimensions randomly and present top 5 entities along those dimensions. As we can see from the results in Table 2, the proposed method produces more coherent entities than the baseline method.

5 Conclusion and Future Works

In this work, we proposed a method for inducing interpretability in KG embeddings using a coherence regularization term. We evaluated the proposed and the baseline method on the interpretability of the learned embeddings. We also evaluated the methods on different KG tasks and compared their performance. We found that the proposed method achieves better interpretability while maintaining comparable performance on KG tasks. As next steps, we plan to evaluate and compare the generalizability of the proposed method across various KG embedding models. Understanding the mapping between dimensions and latent categories could be another direction for future works.

Acknowledgments

We thank the anonymous reviewers for their constructive comments. This work is supported by the Ministry of Human Resources Development (Government of India).

References

Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019. TuckER: Tensor factorization for knowledge graph completion. In *EMNLP-IJCNLP*, pages 5184–5193, Hong Kong, China. ACL.

Federico Bianchi, Gaetano Rossiello, Luca Costabello, Matteo Palmonari, and Pasquale Minervini. 2020. Knowledge graph embeddings and explainable ai. *arXiv preprint arXiv:2004.14843*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Jonathan Chang, Jordan L Boyd-Graber, Sean Gerrish, Chong Wang, and David M Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Nips*, volume 31, pages 1–9.

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, pages 1811–1818.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. 2015. Sparse overcomplete word vector representations. 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 1491–1500, Beijing, China. Association for Computational Linguistics.

Arthur Colombini Gusmao, Alvaro Henrique Chaim Correia, Glauber De Bona, and Fabio Gagliardi Cozman. 2018. Interpreting embedding models of knowledge bases: a pedagogical approach.

Prachi Jain, Sushant Rathi, Soumen Chakrabarti, et al. 2020. Knowledge base completion: Baseline strikes back (again). *arXiv preprint arXiv:2005.00804*.

Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. 2017. Knowledge base completion: Baselines strike back. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 69–74, Vancouver, Canada. Association for Computational Linguistics.

Jey Han Lau, David Newman, and Timothy Baldwin. 2014. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *EACL*, pages 530–539.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed,

N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-ending learning. In *Proceedings of AAAI*.

Brian Murphy, Partha Pratim Talukdar, and Tom Mitchell. 2012. Learning effective and interpretable semantic models using non-negative sparse embedding. In *International Conference on Computational Linguistics (COLING 2012), Mumbai, India*. http://aclweb.org/anthology/C/C12/C12-1118.pdf.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*, volume 11, pages 809–816.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. *NAACL HLT 2013*, pages 74–84.

Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. 2020. You can teach an old dog new tricks! on training knowledge graph embeddings. In *International Conference on Learning Representations*.

M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling. 2017. Modeling Relational Data with Graph Convolutional Networks. *ArXiv e-prints*.

Anant Subramanian, Danish Pruthi, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Eduard Hovy. 2018. Spine: Sparse interpretable neural embeddings. *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *3rd Workshop on Continuous Vector Space Models and Their Compositionality*. ACL – Association for Computational Linguistics.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Lisbon, Portugal. Association for Computational Linguistics.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*.

Han Xiao, Minlie Huang, Yu Hao, and Xiaoyan Zhu. 2015. Transg: A generative mixture model for knowledge graph embedding. *arXiv preprint arXiv:1509.05488*.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. Knowledge semantic representation: A generative model for interpretable knowledge graph embedding. *arXiv preprint arXiv:1608.07685*.

Qizhe Xie, Xuezhe Ma, Zihang Dai, and Eduard Hovy. 2017. An interpretable knowledge transfer model for knowledge base completion. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 950–962, Vancouver, Canada. Association for Computational Linguistics.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Solving Arithmetic Word Problems with Transformers and Preprocessing of Problem Text

Kaden Griffith and Jugal Kalita
University of Colorado
1420 Austin Bluffs Parkway
Colorado Springs CO 80918
kadengriffith@gmail.com and jkalita@uccs.edu

Abstract

This paper outlines the use of Transformer networks trained to translate math word problems to equivalent arithmetic expressions in infix, prefix, and postfix notations. We compare results produced by many neural configurations and find that most configurations outperform previously reported approaches on three of four datasets with significant increases in accuracy of over 20 percentage points. The best neural approaches boost accuracy by 30% when compared to the previous state-of-the-art on some datasets.

1 Introduction

Students are exposed to simple arithmetic word problems starting in elementary school, and most become proficient in solving them at a young age. However, it has been challenging to write programs to solve such elementary school level problems well. Even simple word problems, consisting of only a few sentences, can be challenging to understand for an automated system.

Solving a math word problem (MWP) starts with one or more sentences describing a transactional situation. The sentences are usually processed to produce an arithmetic expression. These expressions then may be evaluated to yield a numerical value as an answer to the MWP.

Recent neural approaches to solving arithmetic word problems have used various flavors of recurrent neural networks (RNN) and reinforcement learning. However, such methods have had difficulty achieving a high level of generalization. Often, systems extract the relevant numbers successfully but misplace them in the generated expressions. They also get the arithmetic operations wrong. The use of infix notation also requires pairs of parentheses to be placed and balanced correctly, bracketing the right numbers. There have been problems with parentheses placement.

Figure 1: Possible generated expressions for an MWP.

Question:
At the fair Adam bought 13 tickets. After riding the ferris wheel he had 4 tickets left. If each ticket cost 9 dollars, how much money did Adam spend riding the ferris wheel?

Some possible expressions that can be produced:
$(13-4)*9, 9*13-4, 5*13-4, 13-4*9, 13-(4*9)$, $(9*13-4), (9)*13-4, (9)*13-(4)$, etc.

To start, correctly extracting the numbers in the problem is necessary. Figure 1 gives examples of some infix representations that a machine learning solver can potentially produce from a simple word problem using the correct numbers. Of the expressions shown, only the first one is correct. The use of infix notation may itself be a part of the problem because it requires the generation of additional characters, the open and closed parentheses, which must be placed and balanced correctly.

The actual numbers appearing in MWPs vary widely from problem to problem. Real numbers take any conceivable value, making it almost impossible for a neural network to learn representations for them. As a result, trained programs sometimes generate expressions that have seemingly random numbers. For example, in some runs, a trained program could generate a potentially inexplicable expression such as $(25.01 - 4) * 9$ for the problem given in Figure 1, with one or more numbers not in the problem sentences. To obviate this issue, we replace the numbers in the problem statement with generic tags like $\langle \mathrm{i} \rangle$, $\langle \mathrm{q} \rangle$, and $\langle \mathrm{x} \rangle$ and save their values as a preprocessing step. This approach does not take away from the generality of the solution but suppresses fertility in number generation leading to the introduction of numbers not present in the question sentences. We extend the prepro-

Proceedings of the 17th International Conference on Natural Language Processing, pages 76–84
Patna, India, December 18 - 21, 2020.

cessing methods to ease the understanding through simple expanding and filtering algorithms. For example, some keywords within sentences are likely to cause the choice of operators for us as humans. By focusing on these terms in the context of a word problem, we hypothesize that our neural approach will improve further.

In this paper, we use the Transformer model (Vaswani et al., 2017) to solve arithmetic word problems as a particular case of machine translation from text to the language of arithmetic expressions. Transformers in various configurations have become a staple of NLP in the past three years. We do not augment the neural architectures with external modules such as parse trees or deep reinforcement learning. We compare performance on four individual datasets. In particular, we show that our translation-based approach outperforms state-of-the-art results reported by (Wang et al., 2018; Hosseini et al., 2014; Kushman et al., 2014; Roy et al., 2015; Robaidek et al., 2018) by a large margin on three of four datasets tested. On average, our best neural architecture outperforms previous results by almost 10%, although our approach is conceptually more straightforward.

We organize our paper as follows. The second section presents related work. Then, we discuss our approach. We follow by an analysis of baseline experimental results and compare them to those of other recent approaches. We then take our best performing model and train it with motivated preprocessing, discussing changes in performance. We follow with a discussion of our successes and shortcomings. Finally, we share our concluding thoughts and end with our direction for future work.

2 Related Work

Past strategies have used rules and templates to match sentences to arithmetic expressions. Some such approaches seemed to solve problems impressively within a narrow domain but performed poorly otherwise, lacking generality (Bobrow, 1964; Bakman, 2007; Liguda and Pfeiffer, 2012; Shi et al., 2015). Kushman et al. (Kushman et al., 2014) used feature extraction and template-based categorization by representing equations as expression forests and finding a close match. Such methods required human intervention in the form of feature engineering and the development of templates and rules, which is not desirable for expandability and adaptability. Hosseini et al. (Hosseini et al., 2014) performed statistical similarity analysis to obtain acceptable results but did not perform well with texts that were dissimilar to training examples.

Existing approaches have used various forms of auxiliary information. Hosseini et al. (Hosseini et al., 2014) used verb categorization to identify important mathematical cues and contexts. Mitra and Baral (Mitra and Baral, 2016) used predefined formulas to assist in matching. Koncel-Kedziorski et al. (Koncel-Kedziorski et al., 2015) parsed the input sentences, enumerated all parses, and learned to match, requiring expensive computations. Roy and Roth (Roy and Roth, 2017) performed searches for semantic trees over large spaces.

Some recent approaches have transitioned to using neural networks. Semantic parsing takes advantage of RNN architectures to parse MWPs directly into equations, or expressions in a math-specific language (Shi et al., 2015; Sun et al., 2019). RNNs have shown promising results, but they have had difficulties balancing parentheses. Sometimes RNN models incorrectly choose numbers when generating equations. Rehman et al. (Rehman et al., 2019) used part-of-speech tagging and classification of equation templates to produce systems of equations from third-grade level MWPs. Most recently, Sun et al. (Sun et al., 2019) used a bi-directional LSTM architecture for math word problems. Huang et al. (Huang et al., 2018) used a deep reinforcement learning model to achieve character placement in both seen and new equation templates. Wang et al. (Wang et al., 2018) also used deep reinforcement learning. We take a similar approach to (Wang et al., 2019) in preprocessing to prevent ambiguous expression representation.

This paper builds upon (Griffith and Kalita, 2019), extending the capability of similar Transformer networks and solving some common issues found in translations. Here, we simplify the tagging technique used in (Griffith and Kalita, 2019), and apply preprocessing to enhance translations.

3 Approach

We view math word problem solving as a sequence-to-sequence translation problem. RNNs have excelled in sequence-to-sequence problems such as translation and question answering. The introduction of attention mechanisms has improved the performance of RNN models. Vaswani et al. (Vaswani et al., 2017) introduced the Transformer network, which uses stacks of attention layers instead of

recurrence. Applications of Transformers have achieved state-of-the-art performance in many NLP tasks. We use this architecture to produce character sequences that are arithmetic expressions. The models we experiment with are easy and efficient to train, allowing us to test several configurations for a comprehensive comparison. We use several configurations of Transformer networks to learn the prefix, postfix, and infix notations of MWP equations independently.

Prefix and postfix representations of equations do not contain parentheses, which has been a source of confusion in some approaches. If the learned target sequences are simple, with fewer characters to generate, it is less likely to make mistakes during generation. Simple targets also may help the learning of the model to be more robust.

3.1 Data

We work with four individual datasets. The datasets contain addition, subtraction, multiplication, and division word problems.

1. **AI2** (Hosseini et al., 2014). AI2 is a collection of 395 addition and subtraction problems containing numeric values, where some may not be relevant to the question.

2. **CC** (Roy and Roth, 2015). The Common Core dataset contains 600 2-step questions. The Cognitive Computation Group at the University of Pennsylvania[1] gathered these questions.

3. **IL** (Roy et al., 2015). The Illinois dataset contains 562 1-step algebra word questions. The Cognitive Computation Group compiled these questions also.

4. **MAWPS** (Koncel-Kedziorski et al., 2016). MAWPS is a relatively large collection, primarily from other MWP datasets. MAWPS includes problems found in AI2, CC, IL, and other sources. We use 2,373 of 3,915 MWPs from this set. The problems not used were more complex problems that generate systems of equations. We exclude such problems because generating systems of equations is not our focus.

We take a randomly sampled 95% of examples from each dataset for training. From each dataset, MWPs not included in training make up the testing data used when generating our results. Training and testing are repeated three times, and reported results are an average of the three outcomes.

[1] https://cogcomp.seas.upenn.edu/page/demos/

3.2 Representation Conversion

We take a simple approach to convert infix expressions found in the MWPs to the other two representations. Two stacks are filled by iterating through string characters, one with operators found in the equation and the other with the operands. From these stacks, we form a binary tree structure. Traversing an expression tree in preorder results in a prefix conversion. Post-order traversal gives us a postfix expression. We create three versions of our training and testing data to correspond to each type of expression. By training on different representations, we expect our test results to change.

3.3 Metric Used

We calculate the reported results, here and in later sections as:

$$model_{avg} = \frac{1}{R}\sum_{r=1}^{R}\left(\frac{1}{N}\sum_{n=1}^{N}\frac{C \in D_n}{P \in D_n}\right) \quad (1)$$

where R is the number of test repetitions, which is 3; N is the number of test datasets, which is 4; P is the number of MWPs; C is the number of correct equation translations, and D_n is the nth dataset.

4 Experiment 1: Search for High-Performing Models

The input sequence for a translation is a natural language specification of an arithmetic word problem. We encode the MWP questions and equations using the subword text encoder provided by the TensorFlow Datasets library. The output is an expression in prefix, infix, or postfix notation, which then can be manipulated further and solved to obtain a final answer. Each expression style corresponds to a model trained and tested separately on that specific style. For example, data in prefix will not intermix with data in postfix representation.

All examples in the datasets contain numbers, some of which are unique or rare in the corpus. Rare terms are adverse for generalization since the network is unlikely to form good representations for them. As a remedy to this issue, our networks do not consider any relevant numbers during training. Before the networks attempt any translation, we preprocess each question and expression by a

number mapping algorithm. We consider numbers found to be in word form also, such as "forty-two" and "dozen." We convert these words to numbers in all questions (e.g., "forty-two" becomes "42"). Then by algorithm, we replace each numeric value with a corresponding identifier (e.g., $\langle j \rangle$, $\langle x \rangle$) and remember the necessary mapping. We expect that this approach may significantly improve how networks interpret each question. When translating, the numbers in the original question are tagged and cached. From the encoded English and tags, a predicted sequence resembling an expression presents itself as output. Since each network's learned output resembles an arithmetic expression (e.g., $\langle j \rangle + \langle x \rangle * \langle q \rangle$), we use the cached tag mapping to replace the tags with the corresponding numbers and return a final mathematical expression.

We train and test three representation models: Prefix-Transformer, Postfix-Transformer, and Infix-Transformer. For each experiment, we use representation-specific Transformer architectures. Each model uses the Adam optimizer with $beta_1 = 0.95$ and $beta_2 = 0.99$ with a standard epsilon of $1 \times e^{-9}$. The learning rate is reduced automatically in each training session as the loss decreases. Throughout the training, each model respects a 10% dropout rate. We employ a batch size of 128 for all training. Each model is trained on MWP data for 300 iterations before testing. The networks are trained on a machine using 4 Nvidia 2080 Ti graphics processing unit (GPU).

We compare medium-sized, small, and minimal networks to show if a smaller network size can increase training and testing efficiency while retaining high accuracy. Networks over six layers have shown to be non-effective for this task. We tried many configurations of our network models but report results with only three configurations of Transformer.

- **Transformer Type 1:** This network is a small to medium-sized network consisting of 4 Transformer layers. Each layer utilizes 8 attention heads with a depth of 512 and a feed-forward depth of 1024.

- **Transformer Type 2:** The second model is small in size, using 2 Transformer layers. The layers utilize 8 attention heads with a depth of 256 and a feed-forward depth of 1024.

- **Transformer Type 3:** The third type of model is minimal, using only 1 Transformer layer. This network utilizes 8 attention heads with a depth of 256 and a feed-forward depth of 512.

Objective Function We calculate the loss in training according to a mean of the sparse categorical cross-entropy formula. Sparse categorical cross-entropy (De Boer et al., 2005) is used for identifying classes from a feature set, assuming a large target classification set. The performance metric evaluates the produced class (predicted token) drawn from the translation classes (all vocabulary subword tokens). During each evaluation, target terms are masked, predicted, and then compared to the masked (known) value. We adjust the model's loss according to the mean of the translation accuracy after predicting every determined subword in a translation.

4.1 Experiment 1 Results

This experiment compares our networks to recent previous work. We count a given test score by a simple "correct versus incorrect" method. The answer to an expression directly ties to all of the translation terms being correct, which is why we do not consider partial precision. We compare average accuracies over 3 test trials on different randomly sampled test sets from each MWP dataset. This calculation more accurately depicts the generalization of our networks.

We present the results of our various outcomes in Table 1. We compare the three representations of target equations and three architectures of the Transformer model in each test.

4.1.1 Experiment 1 Analysis

All of the network configurations used were very successful for our task. The prefix representation overall provides the most stable network performance. We note that while the combined averages of the prefix models outperformed postfix, the postfix representation Transformer produced the highest average for a single model. The type 2 postfix Transformer received the highest testing average of 87.2%. To highlight the capability of our most successful model (type 2 postfix Transformer), we present some outputs of the network in Figure 2.

The models respect the syntax of math expressions, even when incorrect. For most questions, our translators were able to determine operators based solely on the context of language.

Table 1: Test results for Experiment 1 (* denotes averages on present values only).

(Type) Model	AI2	CC	IL	MAWPS	Average
(Hosseini et al., 2014)	77.7	–	–	–	*77.7
(Kushman et al., 2014)	64.0	73.7	2.3	–	*46.7
(Roy et al., 2015)	–	–	52.7	–	*52.7
(Robaidek et al., 2018)	–	–	–	62.8	*62.8
(Wang et al., 2018)	**78.5**	75.5	73.3	–	*75.4
(1) Prefix-Transformer	71.9	**94.4**	**95.2**	83.4	86.3
(1) Postfix-Transformer	73.7	81.1	92.9	75.7	80.8
(1) Infix-Transformer	77.2	73.3	61.9	56.8	67.3
(2) Prefix-Transformer	71.9	**94.4**	94.1	**84.7**	86.3
(2) Postfix-Transformer	77.2	**94.4**	94.1	83.1	**87.2**
(2) Infix-Transformer	77.2	76.7	66.7	61.5	70.5
(3) Prefix-Transformer	71.9	93.3	**95.2**	84.1	86.2
(3) Postfix-Transformer	77.2	94.4	94.1	82.4	87.0
(3) Infix-Transformer	77.2	76.7	66.7	62.4	70.7

Figure 2: Successful postfix translations.

AI2

A spaceship traveled 0.5 light-year from earth to planet x and 0.1 light-year from planet x to planet y. Then it traveled 0.1 light-year from planet y back to Earth. How many light-years did the spaceship travel in all?

Translation produced:

0.5 0.1 + 0.1 +

CC

There were 16 friends playing a video game online when 7 players quit. If each player left had 8 lives, how many lives did they have total?

Translation produced:

8 16 7 - *

IL

Lisa flew 256 miles at 32 miles per hour. How long did Lisa fly?

Translation produced:

256 32 /

MAWPS

Debby's class is going on a field trip to the zoo. If each van can hold 4 people and there are 2 students and 6 adults going, how many vans will they need?

Translation produced:

2 6 + 4 /

Table 1 provides detailed results of Experiment 1. The numbers are absolute accuracies, i.e., they correspond to cases where the arithmetic expression generated is 100% correct, leading to the correct numeric answer. Results by (Wang et al., 2018; Hosseini et al., 2014; Roy et al., 2015; Robaidek et al., 2018) are sparse but indicate the scale of success compared to recent past approaches. Prefix, postfix, and infix representations in Table 1 show that network capabilities are changed by how teachable the target data is.

While our networks fell short of Wang, et al. AI2 testing accuracy (Wang et al., 2018), we present state-of-the-art results for the remaining three datasets in Table 1. The AI2 dataset is tricky because its questions contain numeric values that are extraneous or irrelevant to the actual computation, whereas the other datasets have only relevant numeric values. Following these observations, we continue to more involved experiments with only the type 2 postfix Transformer. The next sections will introduce our preprocessing methods. Note that we start from scratch in our training for all experiments following this section.

5 Experiment 2: Preprocessing for Improved Results

We use various additional preprocessing methods to improve the training and testing of MWP data. One goal of this section is to improve the notably low performance on the AI2 tests. We introduce eight techniques for improvement and report our results as an average of 3 separate training and

testing sessions. These techniques are also tested together in some cases to observe their combined effects.

5.1 Preprocessing Algorithms

We take note of previous pitfalls that have prevented neural approaches from applying to general MWP question answering. To improve English to equation translation further, we apply some transformation processes before the training step in our neural pipeline. First, we optionally remove all stop words in the questions. Then, again optionally, the words are transformed into a lemma to prevent easy mistakes caused by plurals. These simple transformations can be applied in both training and testing and require only base language knowledge.

We also try minimalistic manipulation approaches in preprocessing and analyze the results. While in most cases, the tagged numbers are relevant and necessary when translating; in some cases, there are tagged numbers that do not appear in equations. To avoid this, we attempt three different methods of preprocessing: Selective Tagging, Exclusive Tagging, and Label-Selective Tagging.

When applying Selective Tagging, we iterate through the words in each question and only replace numbers appearing in the equation with a tag representation (e.g., $\langle \text{j} \rangle$). In this method, we leave the original numeric values in each sentence, which have do not collect importance in network translations. Similarly, we optionally apply Exclusive Tagging, which is nearly identical to Selective Tagging, but instead of leaving the numbers not appearing in the equation, we remove them. These two preprocessing algorithms prevent irrelevant numbers from mistakenly being learned as relevant. These methods are only applicable to training.

It is common in MWPs to provide a label or indication of what a number represents in a question. For example, if we observe the statement "George has 2 peaches and 4 apples. Lauren gives George 5 of her peaches. How many peaches does George have?" we only need to know quantifiers for the label "peach." We consider "peaches" and "apples" as labels for the number tags. For the most basic interpretation of this series of mathematical prose, we know that we are supposed to use the number of peaches that George and Lauren have to formulate an expression to solve the MWP. Thus, determining the correct labels or tags for the numbers in an MWP is likely to be helpful. Similar to Selective Tagging and Exclusive Tagging, we avoid tagging irrelevant numbers when applying Label-Selective Tagging. Here, the quantity we need to ignore for a reliable translation is: "4 apples." This is as if we are associating each number with the appropriate unit notation, like "kilogram" or "meter."

The Label-Selective Tagging method iterates through every word in an MWP question. We first count the occurrence of words in the question sentences and create an ordered list of all terms. In our example, we note that the word "peaches" occurs three times in the sentence. Compared to the word "apples" (occurring only once), we reduce our search for numbers to only the most common terms in each question. We impose a check to verify that the most common term appears in the sentence ending in a question mark, and if it is not, the Label-Selective Tagging fails and produces tags for all numbers.

If we can identify the label reliably, we then look at each number. We assume that labels for quantities are within a window of three (either before the number or after). When we detect a number, we then look at four words before the number and four words after the number, and before any punctuation for the most common word we have previously identified. If we find the assumed label, we tag the number, indicating it is relevant. Otherwise, we leave the word as a number, which indicates that it is irrelevant. This method can be applied in training and testing equally because we do not require any knowledge about our target translation equation. We could have performed noun phrase chunking and some additional processing to identify nouns and their numeric quantifiers. However, our heuristic method works very well.

In addition to restrictive tagging methods, we also try replacing all words with an equivalent part-of-speech denotation. The noun, "George," will appear as "NN," for example. There are two ways we employ this method. The first substitutes the word with its part-of-speech counterpart, and the second adds on the part-of-speech tag like "(NN George)," for each word in the sentence. These two algorithms can be applied both in training and testing since the part-of-speech tag comes from the underlying English language.

We also try reordering the sentences in each MWP. For each of the questions, we sort the sentences in random order, not requiring the question within the MWP to appear last, as it typically does.

The situational context is not linear in most cases when applying this transformation. We check to see if the network relies on the linear nature of MWP information to be successful.

The eight algorithms presented are applied solo and in combination, when applicable. For a summary of the preprocessing algorithms, refer to Table 2.

Table 2: Summary of Algorithms.

Name	Abbreviation
Remove Stop Words	SW
Lemmatize	L
Selective Tagging	ST
Label-Selective Tagging	LST
Exclusive Tagging	ET
Part of Speech	POS
Part of Speech w/ Words	WPOS
Sentence Reordering	R

5.1.1 Experiment 2 Results and Analysis

We present the results of Experiment 2 in Table 3. From the results in Table 3, we see that Label-Selective Tagging is very successful in improving the translation quality of MWPs. Other methods, such as removing stop words, improve accuracy due to the reduction in the necessary vocabulary, but fail to outperform LST for three of the four datasets. Using a frequency measure of terms to determine number relevancy is a simple addition to the network training pipeline and significantly outperforms our standalone Transformer network base.

The LST algorithm was successful for two reasons. The first reason LST is more realistic in this application is its applicability to inference time translations. Because the method relies only on each question's vocabulary, there are no restrictions on usability. This method reliably produces the same results in the evaluation as in training, which is a unique characteristic of only a subset of the preprocessing algorithms tested. The second reason LST is better for our purpose is that it prevents unnecessary learning of irrelevant numbers in the questions. One challenge in the AI2 dataset is that some numbers present in the questions are irrelevant to the intended translation. Without some preprocessing, we see that our network sometimes struggles to determine irrelevancy for a given number. LST also reduces compute time for other areas of the data pipeline. Only a fraction of the numbers in some questions need to be tagged for the network, which produces less stress on our number tagging process. Still some common operator inference mistakes were made while using LST, shown in Figure 3.

Figure 3: Example of an unsuccessful translation using type 2 postfix Transformer and LST.

MAWPS

There were 73 bales of hay in the barn. Jason stacked bales in the barn today. There are now 96 bales of hay in the barn . How many bales did he store in the barn?

Translation produced:

96 73 -

Expected translation:

73 96 +

The removal of stop words is a common practice in language classification tasks and was somewhat successful in translation tasks. We see an improvement on all datasets, suggesting that stop words are mostly ignored in successful translations and paid attention to more when the network makes mistakes. There is a significant drop in reliability when we transform words into their base lemmas. Likely, the cause of this drop is the loss of information by imposing two filtering techniques at once.

Along with the successes of the tested preprocessing came some disappointing results. Raw part-of-speech tagging produces slightly improved results from the base model, but including the words and the corresponding part-of-speech denotations fail in our application.

Reordering of the question sentences produced significantly worse results. The accuracy difference mostly comes from the random position of the question, sometimes appearing with no context to the transaction. Some form of limited sentence reordering may improve the results, but likely not the degree of success of the other methods.

By incorporating simple preprocessing techniques, we grow the generality of the Transformer architecture to this sequence-to-sequence task.

5.1.2 Overall Results

Table 3 shows that the Transformer architecture with simple preprocessing outperforms previous state-of-the-art results in all four tested datasets. While the Transformer architecture has been well-proven in other tasks, we show that applying the

Table 3: Test results for Experiment 2 (* denotes averages on present values only). Rows after the top 5 indicate type 2 postfix Transformer results.

Preprocessing Method	AI2	CC	IL	MAWPS	Average
(Hosseini et al., 2014)	77.7	–	–	–	*77.7
(Kushman et al., 2014)	64.0	73.7	2.3	–	*46.7
(Roy et al., 2015)	–	–	52.7	–	*52.7
(Robaidek et al., 2018)	–	–	–	62.8	*62.8
(Wang et al., 2018)	78.5	75.5	73.3	–	*75.4
Type 2 Postfix-Transformer					
None	77.2	94.4	94.1	83.1	87.2
SW	73.7	**100.0**	**100.0**	**94.0**	91.9
L	63.2	86.7	80.9	68.1	74.7
SW + L	61.4	60.0	57.1	66.4	61.2
ST	80.7	93.3	**100.0**	84.3	89.6
SW + ST	71.9	93.3	**100.0**	83.5	87.2
SW + L + ST	50.9	63.3	48.8	64.1	56.8
LST	**82.5**	**100.0**	**100.0**	93.7	**94.0**
SW + LST	70.2	**100.0**	**100.0**	92.6	90.7
SW + L + LST	54.4	71.1	50.0	68.1	60.9
ET	80.7	93.3	**100.0**	84.0	89.5
SW + ET	70.2	93.3	**100.0**	84.0	86.9
SW + L + ET	59.6	63.3	53.6	66.1	60.7
POS	79.0	**100.0**	73.8	91.5	86.1
WPOS	40.4	0.0	84.5	53.0	44.5
R	38.0	59.6	58.7	42.3	49.6

attention schema here improves MWP solving.

With our work, we show that relatively small networks are more stable for our task. Postfix and prefix both are a better choice for training neural networks, with the implication that infix can be re-derived if it is preferred by the user of a solver system. The use of alternative mathematical representation contributes greatly to the success of our translations.

6 Conclusions and Future Work

In this paper, we have shown that the use of Transformer networks improves automatic math word problem-solving. We have also shown that postfix target expressions perform better than the other two expression formats. Our improvements are well-motivated but straightforward and easy to use, demonstrating that the well-acclaimed Transformer architecture for language processing can handle MWPs well, obviating the need to build specialized neural architectures for this task.

In the future, we wish to work with more complex MWP datasets. Our datasets contain basic arithmetic expressions of +, -, *, and /, and only up to 3 of them. For example, datasets such as Dolphin18k (Huang et al., 2016), consisting of web-answered questions from Yahoo! Answers, require a wider variety of language to be understood by the system.

We wish to use other architectures stemming from the base Transformer to maximize the accuracy of the system. For our experiments, we use the 2017 variation of the Transformer (Vaswani et al., 2017), to show that generally applicable neural architectures work well for this task. With that said, we also note the importance of a strategically designed architecture to improve our results.

To further the interest in automatic solving of math word problems, we have released all of the code used on GitHub.[2]

References

Yefim Bakman. 2007. Robust understanding of word problems with extraneous information. *arXiv preprint math/0701393*.

[2] https://github.com/kadengriffith/MWP-Automatic-Solver

Daniel G Bobrow. 1964. *Natural Language Input for a Computer Problem Solving System*. Ph.D. thesis, Massachusetts Institute Of Technology.

Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. 2005. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67.

K. Griffith and J. Kalita. 2019. Solving arithmetic word problems automatically using transformer and unambiguous representations. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 526–532.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533.

Danqing Huang, Jing Liu, Chin-Yew Lin, and Jian Yin. 2018. Neural math word problem solver with reinforcement learning. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 213–223, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 887–896.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. Mawps: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281.

Christian Liguda and Thies Pfeiffer. 2012. Modeling math word problems with augmented semantic networks. In *International Conference on Application of Natural Language to Information Systems*, pages 247–252. Springer.

Arindam Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2144–2153.

Tayyeba Rehman, Sharifullah Khan, Gwo-Jen Hwang, and Muhammad Azeem Abbas. 2019. Automatically solving two-variable linear algebraic word problems using text mining. *Expert Systems*, 36(2):e12358.

Benjamin Robaidek, Rik Koncel-Kedziorski, and Hannaneh Hajishirzi. 2018. Data-driven methods for solving algebra word problems. *arXiv preprint arXiv:1804.10718*.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. *arXiv preprint arXiv:1608.01413*, pages 1743–1752.

Subhro Roy and Dan Roth. 2017. Unit dependency graph and its application to arithmetic word problem solving. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Subhro Roy, Tim Vieira, and Dan Roth. 2015. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics*, 3:1–13.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1132–1142.

Ruiyong Sun, Yijia Zhao, Qi Zhang, Keyu Ding, Shijin Wang, and Cui Wei. 2019. A neural semantic parser for math problems incorporating multi-sentence information. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 18(4):37.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Lei Wang, Dongxiang Zhang, Lianli Gao, Jingkuan Song, Long Guo, and Heng Tao Shen. 2018. Mathdqn: Solving arithmetic word problems via deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Lei Wang, Dongxiang Zhang, Jipeng Zhang, Xing Xu, Lianli Gao, Bing Tian Dai, and Heng Tao Shen. 2019. Template-based math word problem solvers with recursive neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7144–7151.

Clickbait in Hindi News Media : A Preliminary Study

Vivek Kaushal
International Institute of
Information Technology, Hyderabad
India
vivek.kaushal@research.iiit.ac.in

Kavita Vemuri
International Institute of
Information Technology, Hyderabad
India
kvemuri@iiit.ac.in

Abstract

A corpus of Hindi news headlines shared on Twitter was created by collecting tweets of 5 mainstream Hindi news sources for a period of 4 months. 7 independent annotators were recruited to mark the 20 most retweeted news posts by each of the 5 news sources on its clickbait nature. The clickbait score hence generated was assessed for its correlation with interactions on the platform (retweets, favorites, reader replies), tweet word count, and normalized POS (part-of-speech) tag counts in tweets. A positive correlation was observed between readers' interactions with tweets and tweets' clickbait score. Significant correlations were also observed for POS tag counts and clickbait score. The prevalence of clickbait in mainstream Hindi news media was found to be similar to its prevalence in English news media. We hope that our observations would provide a platform for discussions on clickbait in mainstream Hindi news media.

1 Introduction

A news headline provides a brief introduction to the news story and perhaps more importantly, lays emphasis on the focus and scope of the accompanying news article. A common journalistic advice is to present a clear attention-grabbing headline but to not exaggerate and misreport the news story or mislead the reader. With the advent of social media and news aggregators, newsreaders are encountering headlines from a variety of sources – established traditional publishers, up-and-coming online news sources, individual writers, among others – some of which apply questionable tactics to attract the attention of readers. This bombardment of information has led to an overload and hence readers gravitate towards sensationalism as a valid filter (Molek-Kozakowska, 2013).

In the digital media space, catchy headlines that lure readers to click on them and link to accompanying articles are called 'clickbait' (Chakraborty et al., 2016; Chen et al., 2015b). The social media site Facebook has defined clickbait as an implementation of forward referencing – "when a publisher posts a link with a headline that encourages people to click to see more, without telling them much information about what they will see"(El-Arini and Tang, 2014). There is a growing fear that the line between traditional headlines, predatory clickbait and fake news is rapidly blurring (Chen et al., 2015a). While there is agreement that not all catchy headlines are malicious, provided the corresponding article satisfies the information gap in the headline with factual reporting (Rony et al., 2017), the concern raised in journalistic circles is the intent to misuse literary techniques and mislead readers for mere profit. Additionally, one needs to look into whether the traffic generated by clickbait headlines with minimal or untrue information leads to social distrust and affects cognitive processes. That is, are clickbait akin in nature to rumors and half-truths and more importantly, what is the impact of headline-only-readers (as reported by the satirical news site Science Post, which posted a catchy headline with nonsensical article content and found that nearly 46,000 shared the article – without even opening the article linked to the headline)[1] who could propagate misinformation by filling in the knowledge gaps by unverified intuitions and information from uncertified sources.

The existing body of research on clickbait is overwhelmingly focused on the English language, except a few notable studies (Orosa et al., 2017; Gabielkov et al., 2016). This includes research on clickbait's reach, impact and its detection. In this preliminary study, we extend the English language clickbait headlines analysis to Hindi. The interest is to understand whether journalists from non-

[1] http://thesciencepost.com/study-70-of-facebook-commenters-only-read-the-headline/

Proceedings of the 17th International Conference on Natural Language Processing, pages 85–89
Patna, India, December 18 - 21, 2020.

English papers apply clickbait as strategy, as was found in a study of 28 EU news sources (Orosa et al., 2017). By also considering the sharing behavior of headline readers, the aim is to understand relations between the demand-supply of clickbait headlines.

We have reported the findings of a pilot study conducted on news headlines shared by five mainstream Hindi news sources on Twitter. In the country where the study was conducted, approximately 40% of the population's mother tongue is Hindi[2]. All the recruited annotators had Hindi as their native tongue and English as a secondary language. The large and rapidly growing reader-base which is increasingly present on social media, places the relevance of this study.

We have annotated the most retweeted news headlines of the five selected news sources on its clickbait-nature and have done POS tagging of tweets' text. Through a detailed correlation analysis of the 'clickbait-score' generated through manual annotation, with tweets' interaction parameters (likes, retweets, replies), text word count and normalized POS tag counts, we hope to understand the existence and reach of clickbait in Hindi news media. The need to understand reader's reading behavior is important for a deeper discussion on the proliferation of clickbait, as studies have shown the impact of clickbait in causing higher stress levels and lowering productivity (Mark, 2014).

2 Methodology

2.1 Preparing the Corpus

Hindi news publishers with text as their primary medium were selected based on their popularity on social media. The selected news sources included: BBC Hindi, Dainik Jagran, Dainik Bhaskar, Hindustan and Navbharat Times. The tweets from the respective publishers' Twitter handles were scraped for a period of 4 months (May, 2020 till the end of August, 2020). Besides the tweeted text, information was also collected on each tweet's interactions on the platform, quantified by the number of readers' replies, retweets and favorites. The usage of mentions and hashtags in the tweets were also recorded. The scraped tweets were filtered to remove instances of polls, cartoons, images and videos. Tweets that were replies to other users were also excluded from the corpus. After filtering, the generated corpus had around 52,000 tweets of definite news headlines invariably accompanied by a link to a news article on the publisher's website [3]. The corpus was further processed to remove mentions and hashtags used at the end of tweets to promote them on the platform. The few instances where mentions or hashtags were used in running text of headlines were not removed as doing so would have disrupted the semantic structure of the tweet. This corpus was sorted according to the number of retweets and 20 top tweets were selected for each of the 5 news publishers. No particular filter was applied on the topic of the news headline, in this preliminary study.

News Source	Most Retweeted News
BBC Hindi	यूट्यूब पर 'नापसंद' की जा रही है पीएम मोदी के 'मन की बात'
Dainik Jagran	ट्विटर पर प्रधानमंत्री के फॉलोअर्स की संख्या छह करोड़ पहुंचना उनकी बढ़ती लोकप्रियता का प्रमाण है?
Hindustan	#TonyKakkar और #ShehnaazGill का नया गाना कुर्ता पजामा रिलीज हो गया है।
Navbharat Times	बीजेपी नेता छपवा रहा था NCERT की 'नकली किताबें', 35 करोड़ का माल बरामद
Dainik Bhaskar	सोशल मीडिया पर JEE-NEET के खिलाफ कैंपेन: शिक्षामंत्री पर भड़के स्टूडेंट्स बोले- अपनी वेब साइट पर पोल कराकर देख ली जिए, पता चल जाएगा कितने बच्चे परीक्षा देना चाहते हैं

Table 1: Most retweeted news item for each news source in our dataset.

2.2 Clickbait Annotation

7 independent bi-lingual annotators (age: $\mu = 31.6$ years, $\sigma = 13.5$ years) with Hindi as their native tongue and English as a secondary language were provided with standard English definitions and English examples of clickbait and non-clickbait headlines. The annotators were provided with only the Devanagari script text of the 100 most retweeted Hindi news headlines presented in a random order and asked to rate each of them on a 5-point linear scale – [0, 0.25, 0.5, 0.75, 1] where 0 implies not clickbait at all and 1 implies strongly

[2] https://www.thehindu.com/data/what-percentage-of-people-prefer-to-speak-hindi-across-states/article27451589.ece

[3] The dataset of scraped hindi new media tweets is open and available at https://github.com/kaushalvivek/hindi-media-tweets

clickbait. The annotation setup was similar to existing research on English news tweets (Potthast et al., 2018), as standard definitions and examples of clickbait were provided to the annotators before they assigned a clickbait-score to presented headlines based on their own perception. The responses were saved and mean of all the 7 annotators' response was taken to generate a 'clickbait score' for each tweet.

2.3 Data Analysis

Post annotation, correlation of the generated clickbait score with different parameters of tweets, including the number of replies, retweets, favorites and text word count were studied. A POS tagger based on the Hidden Markov Model was used to assign part of speech tags following the Viterbi algorithm (Ekbal et al., 2007). POS tag counts in tweets were normalized for tweets' word count to isolate the relation between POS tags' occurrence and clickbait score. The correlation between normalized POS tag counts and clickbait score was studied and reported.

The D'Agostino-Pearson Test was conducted on all distributions in the dataset to check for normality, a pre-condition for correlation analysis. All the distributions in our assessment were found to be normal, hence Pearson's test was conducted to evaluate correlation between different variables. The number of annotators were low, but the Cronbach's alpha was calculated for the 7 independent annotators to check for internal consistency and the annotations were found to be consistent ($\alpha \geq 0.8$).

3 Results

The average clickbait score for all the annotated Hindi news tweets was $\mu = 0.433$, $\sigma = 0.30$. BBC Hindi had the highest average clickbait score ($\mu = 0.59$, $\sigma = 0.30$), while Dainik Bhaskar had the lowest average clickbait score ($\mu = 0.22$, $\sigma = 0.21$). Clickbait scores of all the news sources in the dataset are illustrated (in figure 1). 21 out of the 100 most retweeted news headlines had a clickbait score ≥ 0.75, whereas 36 out of the 100 most retweeted news headlines had a clickbait score ≤ 0.25.

A positive correlation was observed between clickbait score of tweets and all their interaction parameters on Twitter - replies ($r = 0.25$), retweets ($r = 0.19$) and favorites ($r = 0.18$). While the correlation was strongly negative ($r = -0.39$) between

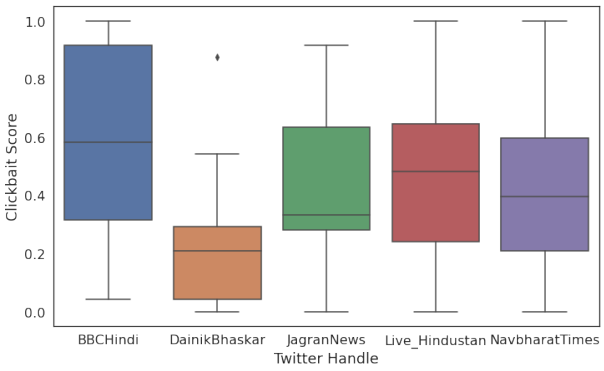


Figure 1: Boxplot of the clickbait score for each news source in our dataset. BBC Hindi ($\mu = 0.59$, $\sigma = 0.30$), Dainik Bhaskar ($\mu = 0.22$, $\sigma = 0.21$), Dainik Jagran ($\mu = 0.44$, $\sigma = 0.27$), Hindustan ($\mu = 0.47$, $\sigma = 0.29$) and Navbharat Times ($\mu = 0.44$, $\sigma = 0.29$).

Tweets' Parameter	**Correlation with Clickbait Score (r)**	**Significance of Correlation (p)**
Replies	0.25	<0.05
Retweets	0.19	0.05
Favorites	0.18	0.07
Word Count	-0.39	<0.001

Table 2: Pearson's r (correlation) between clickbait score and tweets' parameters.

the word count of tweets and their clickbait score. As would make intuitive sense, the interaction parameters in themselves were very strongly correlated with each other ($r \geq 0.7$). The findings are illustrated in table 2.

Upon normalizing POS tag counts in each tweet and assessing its correlation with clickbait score, VAUX ($r = 0.18$), WQ ($r = 0.44$), NNP ($r = 0.17$) and QO ($r = 0.10$) were found to be positively correlated with clickbait score, while INJ ($r = -0.18$), DEM ($r = -0.21$) and SYM ($r = -0.25$) were found to be negatively correlated with clickbait score. No significant correlation was found for other POS tags. The results are illustrated in table 3 in a categorization of POS tags based on correlation with clickbait score.

4 Discussion

Against common perception that clickbait is used largely by fringe players who do not post mainstream news material, Rony et al. (2017) had found that 33.54% of social media posts by mainstream English media was clickbait in nature. The ra-

Correlation	POS Tags
Positive (>0.1)	VAUX, WQ, NNP, QO
Negative (<-0.1)	INJ, DEM, SYM
Not Significant	JJ, RP, PRP, INTF, VM, NEG, NN, RB, QF, NST, PSP, CC, XC, QC

Table 3: Correlation of POS Tags normalized for tweet word-count and clickbait score.

tio was even more alarming at 47.56% for mainstream broadcast media. The trend is reflected in our study, where we found that 21% of the most retweeted news headlines by mainstream Hindi print news media had a high clickbait score (clickbait score ≥ 0.75), which is fairly close to the 24.12% clickbait-content ratio for English print news media found by Rony et al. (2017). The similarities between the ratio of clickbait content posted by mainstream English and Hindi news publishers is an indication that proliferation of clickbait is not limited to English. This points to a need to replicate suitable measures for AI-based detection and management of clickbait as are being developed for the English language (Chakraborty et al., 2016; Agrawal, 2016; Biyani et al., 2016; Zhou, 2017; Venneti and Alam, 2018).

Clickbait score for our Hindi corpus was positively correlated with interaction parameters on Twitter, hence indicating that clickbait content is shared more widely and attracts higher reader attention. This observation is in agreement with existing clickbait research in the English language(Chakraborty et al., 2017) . A strong negative correlation observed between clickbait-score and word-count indicates that clickbait headlines in Hindi are shorter than traditional non-clickbait headline. This is an interesting result that needs to be further examined to isolate effects from language-structure, as similar research studies in the English language have found non-clickbait headlines to be significantly shorter than clickbait headlines (Chakraborty et al., 2016).

The strong positive correlation observed between normalized WQ (question word) counts and clickbait score is expected, as forward-reference – a clickbait technique which involves the omission of a key piece of information in the headline, frequently relies on framing headlines as questions. E.g. "पाकिस्तान में 15 रुपए सस्ता हुआ पेट्रोल, भारत में क्यों नहीं?" – which translates to "Petrol prices reduced by Rs.15 in Pakistan, why not in India?" This headline does not provide any information on why petrol prices are not being reduced in India but generates curiosity by framing the headline as a question, and possibly the emotional/political connotations of the countries being compared. A positive correlation was observed between normalized counts of VAUX (Auxiliary Verb), NNP (Proper Noun), QO (Ordinals) and clickbait score, and a negative correlation was found between INJ (Interjection), DEM (Demonstrative), SYM (Symbol) and clickbait score. These findings require further study and analysis to fully understand the role of each POS tag in evoking curiosity and grabbing attention. Journalistic guidelines suggest that auxiliary verbs are not necessary for perfect passive structures, a proper noun contextualizes and ordinals convey a position or rank which are recommended. Hence, the positive correlation of the two latter POS with clickbait scores is explicable. The negative correlation with interjection is unanticipated as emotional valence attached is considered to be a strong bait for human attention. The similarities between the prevalence of clickbait content in mainstream Hindi news media with recorded observations in English news media is insightful as the research to counter such content in Hindi is fairly primitive. The authors of this paper are working to understand the impact of clickbait on the credibility of news media as perceived by readers (Kaushal and Vemuri, 2020) and plan on scaling their work to news headlines in the Hindi language. We hope that our work would provide a platform for discussion about clickbait in mainstream Hindi news media.

5 Limitations

The preliminary study and the inferences drawn is limited by the low number of annotators (7) and the filter applied to study only 100 headlines. Further, cross-reference to article content of a headline was not included in this study, thus limiting the scores from only the headlines. The length of the headlines was not controlled, though not shown to affect the clickbait score, it could weigh the attention score.

References

A. Agrawal. 2016. Clickbait detection using deep learning. In *2016 2nd International Conference on Next*

Generation Computing Technologies (NGCT), pages 268–272.

Prakhar Biyani, Kostas Tsioutsiouliklis, and John Blackmer. 2016. "8 amazing secrets for getting more clicks": Detecting clickbaits in news streams using article informality. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 94–100. AAAI Press.

A. Chakraborty, B. Paranjape, S. Kakarla, and N. Ganguly. 2016. Stop clickbait: Detecting and preventing clickbaits in online news media. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 9–16.

Abhijnan Chakraborty, Rajdeep Sarkar, Ayushi Mrigen, and Niloy Ganguly. 2017. Tabloids in the era of social media? understanding the production and consumption of clickbaits in twitter. *Proc. ACM Hum.-Comput. Interact.*, 1(CSCW).

Yimin Chen, Nadia K. Conroy, and Victoria L. Rubin. 2015a. News in an online world: The need for an "automatic crap detector". *Proceedings of the Association for Information Science and Technology*, 52(1):1–4.

Yimin Chen, Niall J. Conroy, and Victoria L. Rubin. 2015b. Misleading online content: Recognizing clickbait as "false news". In *Proceedings of the 2015 ACM on Workshop on Multimodal Deception Detection*, WMDD '15, page 15–19, New York, NY, USA. Association for Computing Machinery.

Asif Ekbal, Samiran Mandal, and Sivaji Bandyopadhyay. 2007. Pos tagging using hmm and rule-based chunking. *Shallow Parsing for South Asian Languages*, page 25.

Khalid El-Arini and Joyce Tang. 2014. Click-baiting. *About Facebook*.

Maksym Gabielkov, Arthi Ramachandran, Augustin Chaintreau, and Arnaud Legout. 2016. Social clicks: What and who gets read on twitter? In *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science*, SIGMETRICS '16, page 179–192, New York, NY, USA. Association for Computing Machinery.

Vivek Kaushal and Kavita Vemuri. 2020. Clickbait - trust and credibility of digital news. IEEE International Symposioum on Technology and Society (under review, accepted with changes).

Gloria Mark. 2014. You won't believe what happened next. *The New York Times*.

Katarzyna Molek-Kozakowska. 2013. Towards a pragma-linguistic framework for the study of sensationalism in news headlines. *Discourse amp Communication*, 7:173–197.

B Gracia Orosa, S Gallur Santorun, and X Lopez Gracia. 2017. Use of clickbait in the online news media of the 28 eu member countries. *Revista Latina de Comunicación Social*, 72:1.261–1.277.

Martin Potthast, Tim Gollub, Matthias Hagen, and Benno Stein. 2018. The clickbait challenge 2017: Towards a regression model for clickbait strength. *CoRR*, abs/1812.10847.

Md Main Uddin Rony, Naeemul Hassan, and Mohammad Yousuf. 2017. Diving deep into clickbaits: Who use them to what extents in which topics with what effects? In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, ASONAM '17, page 232–239, New York, NY, USA. Association for Computing Machinery.

Lasya Venneti and Aniket Alam. 2018. How curiosity can be modeled for a clickbait detector. *CoRR*, abs/1806.04212.

Yiwei Zhou. 2017. Clickbait detection in tweets using self-attentive network. *CoRR*, abs/1710.05364.

Self Attended Stack Pointer Networks for Learning Long Term Dependencies

Salih Tuç
Hacettepe University
Department of Computer Engineering
Ankara, Turkey
salihtuc0@gmail.com

Burcu Can
University of Wolverhampton
Research Institute of
Information and Language Processing
Wolverhampton, UK
b.can@wlv.ac.uk

Abstract

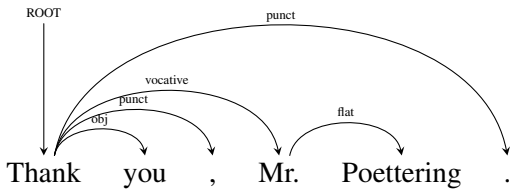
We propose a novel deep neural architecture for dependency parsing, which is built upon a Transformer Encoder (Vaswani et al., 2017) and a Stack Pointer Network (Ma et al., 2018). We first encode each sentence using a Transformer Network and then the dependency graph is generated by a Stack Pointer Network by selecting the head of each word in the sentence through a head selection process. We evaluate our model on Turkish and English treebanks. The results show that our trasformer-based model learns long term dependencies efficiently compared to sequential models such as recurrent neural networks. Our self attended stack pointer network improves UAS score around 6% upon the LSTM based stack pointer (Ma et al., 2018) for Turkish sentences with a length of more than 20 words.

1 Introduction

Dependency Parsing is the task of finding the grammatical structure of a sentence by identifying syntactic and semantic relationships between words. Dependency parsing has been utilized in many other NLP tasks such as machine translation (Carreras and Collins, 2009; Chen et al., 2017), relation extraction (Fundel-Clemens et al., 2007; Zhang et al., 2018), named entity recognition (Jie et al., 2017; Finkel and Manning, 2009), information extraction (Angeli et al., 2015; Peng et al., 2017), all of which involve natural language understanding to an extent. Each dependency relation is identified between a head word and a dependent word that modifies the head word in a sentence. Although such relations are considered syntactic, they are naturally built upon semantic relationships between words. For example, each dependent has a role in modifying its head word, which is a result of a semantic influence.

Within the context of dependency parsing, relations between heads and dependents are also labeled by specifying the type of the grammatical relation between words. In the Universal Dependencies (de Marneffe et al., 2014) tagset, there are 37 dependency relation types defined. In the latest Universal Dependencies (UD v2.0) tagset, relations are split into four main categories (Core Arguments, Non-core dependents, Nominal dependents and Other) and nine sub-categories (Nominals, Clauses, Modifier Words, Function Words, Coordination, MWE, Loose Special and Other).

One way to illustrate the grammatical structure obtained from dependency parsing is a dependency graph. An example dependency graph is given below:



Here, the relations are illustrated by the links from head words to dependent words along with their dependency labels. Every sentence has a global head word, which is the *ROOT* of the sentence.

There are two main difficulties in dependency parsing. One is the long term dependencies in especially long sentences that are difficult to be identified in a standard Recurrent Neural Network due to the loss of the information flow in long sequences. Another difficulty in parsing is the out-of-vocabulary (OOV) words. In this work, we try to tackle these two problems by using Transformer Networks (Vaswani et al., 2017) by introducing subword information for OOV words in especially morphologically rich languages such as Turkish. For that purpose, we integrate character-level word embeddings obtained from Convolutional Neural Networks (CNNs). The morphological complexity

Proceedings of the 17th International Conference on Natural Language Processing, pages 90–100
Patna, India, December 18 - 21, 2020.

in such agglutinative languages makes the parsing task even harder because of the sparsity problem due to the number of suffixes that each word can take, which brings more problems in syntactic parsing. Dependencies in such languages were also defined between morphemic units (i.e. inflectional groups) rather than word tokens (Eryiğit et al., 2008), however this is not in the scope of this work.

In this work, we introduce a novel two-level deep neural architecture for graph-based dependency parsing. Graph-based dependency parsers build dependency trees among all possible trees, therefore the final dependency tree has the highest score globally. However, in transition-based dependency parsers, each linear selection in a sentence is made based on a local score which may lead to erroneous trees at the end of parsing. For this reason, we prefer graph-based dependency parsing in our approach to be able to do global selections while building dependency trees. In the first level of our deep neural architecture, we encode each sentence through a transformer network (Vaswani et al., 2017), which shows superior performance in long sequences compared to standard recurrent neural networks (RNNs). In the second level, we decode the dependencies between heads and dependents using a Stack Pointer Network (Ma et al., 2018), which is extended with an internal stack based on pointer networks (Vinyals et al., 2015). Since stack pointer networks benefit from the full sequence similar to self attention mechanism in transformer networks, they do not have left-to-right restriction as in transition based parsing. Hence, we combine the two networks to have a more accurate and efficient dependency parser.

We evaluate our model on Turkish which is a morphologically rich language and on English with a comparably poorer morphological structure. Although our model does not outperform other recent model, it shows competitive performance among other neural dependency parsers. However, our results show that our self attended stack pointer network improves UAS score around 6% upon the LSTM based stack pointer (Ma et al., 2018) for Turkish sentences with a length of more than 20 words.

The paper is organized as follows: Section 2 reviews the related work on both graph-based and transition-based dependency parsing, Section 3 explains the dependency parsing task briefly, Section 4 describes the proposed deep neural architecture based on Transformer Networks and Stack Pointer Networks, and finally Section 5 presents the experimental results of the proposed model for both English and Turkish.

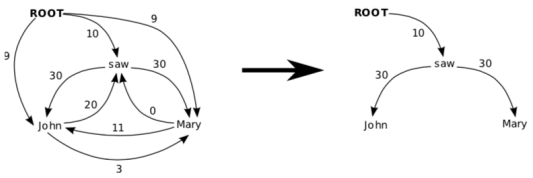


Figure 1: An example to graph-based dependency parsing with a maximum spanning tree.

2 Related Work

Dependency parsing is performed by two different approaches: graph-based and transition-based parsing. We review related work on both of these approaches.

Graph-based Dependency Parsing: Graph-based approaches are generally based on performing the entire parsing process as graph operations where the nodes in the graph represent the words in a sentence. For the sentence, *"John saw Mary"*, we can illustrate its parse tree with a weighted graph G with four vertices where each of them refers to a word including the $ROOT$. Edges store the dependency scores between the words. The main idea here is to find the maximum spanning tree of this graph G. The parse tree of the sentence is given in Figure 1. The dependencies are between $ROOT$ and saw, saw and $John$; and saw and $Mary$ where the first ones are the heads and the latter ones are the dependents.

When the parsing structure is represented as a graph, finding dependencies becomes easier to visualize, and moreover the task becomes finding the highest scored tree among all possible trees. Edge scores in the graphs represent the dependency measures between word couples.

Neural architectures have been used for graph-based dependency parsing extensively in the last decade. Li et al. (2018) introduce a seq2seq model using bi-directional LSTMs (BiLSTMs) (Hochreiter and Schmidhuber, 1997), where an attention mechanism is involved between the encoder and decoder LSTMs. Kiperwasser and Goldberg (2016) propose another model using BiLSTMs, where the right and left arcs in the dependency trees are identified through the BiLSTMs. Dozat and Manning (2016) proposes a parser that uses biaffine attention mechanism, which is extended based on the models

of Kiperwasser and Goldberg (2016), Hashimoto et al. (2017), and Cheng et al. (2016). The biaffine parser (Dozat and Manning, 2016) provides a baseline for other two models introduced by Zhou and Zhao (2019) and Li et al. (2019), which forms trees in the form of Head-Driven Phase Structure Grammar (HPSG) and uses self-attention mechanism respectively. Ji et al. (2019) propose a Graph Neural Network (GNN) that is improved upon the biaffine model. Another LSTM-based model is introduced by Choe and Charniak (2016), where dependency parsing is considered as part of language modelling (LM) and each sentence is parsed with a LSTM-LM architecture which builds parse trees simultaneously with the language model.

The recent works generally focus on the encoder in seq2seq models because a better encoding of an input eliminates most of the cons of the sequence models. For example, Hewitt and Manning (2019) and Tai et al. (2015) aim to improve the LSTM-based encoders while Clark et al. (2018) introduce an attention-based approach to improve encoding, where they propose Cross-View Training (CVT).

In this work, we encode each sentence through a transformer network based on self-attention mechanism (Vaswani et al., 2017) and learn the head of each word using a stack pointer network as a decoder (Ma et al., 2018) in our deep neural architecture. Our main aim is to learn long term dependencies efficiently with a transformer network by removing the recurrent structures from encoder. Transformer networks (Vaswani et al., 2017) and stack pointer networks (Ma et al., 2018) have been used for dependency parsing before. However, this will be the first attempt to combine these two methods for the dependency parsing task.

Transition-based Dependency Parsing: In transition-based dependency parsing, local selections are made for each dependency relationship without considering the complete dependency tree. Therefore, globally motivated selections are normally not performed in transition-based parsing by contrast with graph-based dependency parsing. For this purpose, two stacks are employed to keep track of the actions made during transition-based parsing.

Similar to graph-based parsing, neural approaches have been used extensively for transition-based parsing. Chen and Manning (2014) introduce a feed forward neural network with various extensions by utilizing single-word, word-pair and three-word features. Weiss et al. (2015) improve upon the model by Chen and Manning (2014) with a deeper neural network and with a more structured training and inference using structured perceptron with beam-search decoding. Andor et al. (2016) use also feed forward neural networks similar to others and argue that feed forward neural networks outperform RNNs in case of a global normalization rather than local normalizations as in Chen and Manning (2014), which apply greedy parsing.

Mohammadshahi and Henderson (2019) utilize a transformer network, in which graph features are employed as input and output embeddings to learn graph relations, thereby their novel model, Graph2Graph transformer, is introduced.

Fernández-González and Gómez-Rodríguez (2019) propose a transition-based algorithm that is similar to the stack pointer model by Ma et al. (2018); however, left-to-right parsing is adopted on the contrary to Ma et al. (2018), where top-down parsing is performed. Hence, each parse tree is built in n actions for an n length sentence without requiring any additional data structure.

In addition to these models, there are some works such as the greedy parser of Ballesteros et al. (2016) and Kuncoro et al. (2016), and the high-performance parser by Qi and Manning (2017).

Nivre and McDonald (2008) indicate that graph-based and transition-based parsers can be also combined by integrating their features. And several works follow this idea (Goldberg and Elhadad, 2010; Spitkovsky et al., 2010; Ma et al., 2013; Ballesteros and Bohnet, 2014; Zhang and Clark, 2008).

3 The Formal Definition of Dependency Parsing

Dependency parsing is the task of inferring the grammatical structure of a sentence by identifying the relationships between words. Dependency is a head-dependent relation between words and each *dependent* is affected by its *head*. The dependencies in a dependency tree are always from the head to the dependents.

The parsing, no matter which approach is used, creates a dependency tree or a graph, as we mentioned above. There are some formal conditions of this graph:

- Graph should be connected.
  - Each word must have a head.

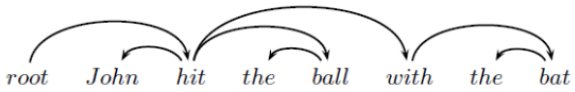


Figure 2: An example projective tree

Figure 3: An example non-projective tree

- Graph must be acyclic.
  - If there are dependencies $w1 \rightarrow w2$ and $w2 \rightarrow w3$; there must not be a dependency such as $w3 \rightarrow w1$.
- Each of the vertices must have one incoming edge.
  - Each word must only have one head. A graph that includes $w1 \rightarrow w2$ and $w3 \rightarrow w2$ is not allowed in a dependency graph.

A dependency tree is projective if there are no crossing edges on the dependency graph. Figure 2 illustrates a projective tree and Figure 3 illustrates a non-projective dependency graph.

4 Dependency Parsing with Self Attended Stack Pointer Network

4.1 Overview

Self Attended Stack Pointer Network is extended on a standard Stack Pointer Network (STACKPTR) (Ma et al., 2018) along with a self attention mechanism. In STACKPTR, input word embeddings are processed via a BiLSTM-CNN encoder, where a BiLSTM is utilized to encode each word and a CNN is utilized to learn character-based encoding of each word. All words are stored in a stack structure and each encoded word on the top of the stack is decoded using an LSTM decoder to discover their heads by utilizing high-order information such as siblings and grandparents. Finally, each dependency relation is predicted through a Deep BiAffine Parser Dozat and Manning (2016) in a standard Pointer Network architecture.

Our model deviates from the STACKPTR model with a transformer network that encodes each word with a self-attention mechanism, which will allow to learn long-term dependencies since every word's relation to all words in a sentence can be effectively processed in a transformer network on the contrary to recurrent neural networks. In sequential recurrent structures such as RNNs or LSTMs, every word's encoding contains information about only previous words in a sentence and there is always a loss in the information flow through the long sequences in those structures.

In our transformer network, we adopt a multi-head attention and a feed-forward network. Once we encode a sequence with a transformer network, we decode the sequence to predict the head of each word in that sequence by using a stack pointer network.

4.2 Transformer Encoder

In RNNs, each state is informed by the previous states with a sequential information flow through the states. However, in longer sequences, information passed from earlier states loses its effect on the later states in RNNs by definition. Transformer networks are effective attention-based neural network architectures (Vaswani et al., 2017). The main idea is to replace the recurrent networks with a single transformer network which has the ability to compute the relationships between all words in a sequence with a self-attention mechanism without requiring any recurrent structure. Therefore, each word in a sequence will be informed by all other words in the sequence.

Learning long term dependencies in especially long sentences is still one of the challenges in dependency parsing. We employ transformer networks in order to tackle with the long term dependencies problem by eliminating the usage of recurrent neural networks while encoding each sentence during parsing. Hence, we use transformer network as an encoder to encode each word by feeding our transformer encoder with each word's pretrained word embeddings (Glove (Pennington et al., 2014) or Polyglot (Al-Rfou' et al., 2013) embeddings), part-of-speech (PoS) tag embeddings, character-level word embeddings obtained from CNN, and the positional encodings of each word.

Positional encoding (PE) is used to inject positional information for each encoded word, since there is not a sequential recurrent structure in a self attention mechanism. With the positional encoding, some relative or absolute positions of words in a sentence are utilized. The cos function is used for the odd indices and the sin function is used for even indices. The injection of the position information is performed with the sinus waves. The

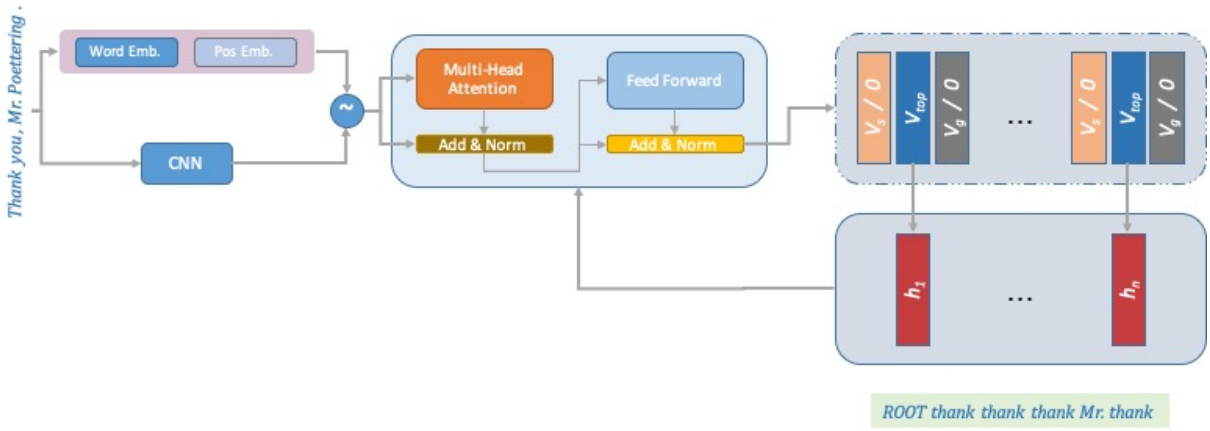


Figure 4: Overview of the Self-Attended Pointer Network Model. After concatenating word embeddings, POS tag embeddings, and char-embeddings obtained from CNN, the final embedding is fed into the self-attention encoder stack. Then, embedding of the word at the top of the stack, its sibling and grandparent vectors are summed-up in order to predict the dependency head.

sin function for the even indices is computed as follows:

$$PE(x, 2i) = \sin\left(\frac{x}{10000^{2i/d_{model}}}\right) \quad (1)$$

where d_{model} is the dimension of the word embeddings, $i \in [0, d_{model}/2)$, and x is the position of each word where $x \in [0, n]$ in the input sequence $s = (w_0, w_1 \dots w_n)$. The cos function for the odd indices is computed analogously.

The positional encoding is calculated for each embedding and they are summed. So the dimension d_{model} does not change. Concatenation is also possible theoretically. However, in the input and output embeddings, the position information is included in the first few indices in the embedding. Thus, when the d_{model} is large enough, there is no need to concatenate. The summation also meets the requirements.

The Encoder stack contains a Multi-Head Attention and a Feed-Forward Network. A Layer Normalization is applied after each of these two layers. There could be more than one encoder in the encoder stack. In this case, all of the outputs in one encoder is fed into the next encoder in the encoder stack. In our model, we performed several experiments with different number of encoder layers in the encoder stack to optimize the number of encoder layers for parsing.

Multi-Head Attention is evolved from Self-Attention Mechanism, which enables encoding all words using all of the words in the sentence. So it learns better relations between words compared to recurrent structures. The all-to-all encoding in self-attention mechanism is performed through query, key and value matrices. There are multiple sets of queries, keys and values that are learned in the model. Self-attention is calculated for each of these sets and a new embedding is produced. The new embeddings for each set are concatenated and multiplied with Z matrix which is a randomly-initialized matrix in order to compute the final embeddings. Z matrix is trained jointly and multiplied with the concatenated weight matrix in order to reduce the embeddings into a single final embedding for each set. In other words, the final embedding is learnt from different contexts at the same time. It is multi-head because it learns from the *head* of each set. The *head* of each set is calculated by using self-attention.

Finally, a Feed Forward Neural Network which is basically a neural network with two linear layers and ReLU activation function is used to process the embeddings obtained from multi-head attention. It is placed at the end of the encoder because with this feed-forward neural network, we can train the embeddings with a latent space of words.

Layer Normalization (Ba et al., 2016) is applied to normalize the weights and retain some form of information from the previous layers, which is performed for both Multi-Head Attention and Feed Forward Neural Network.

Final output embeddings contain contextual information about the input sentence and the words in the sequence. So, the output of the Transformer Encoder is a -theoretically- more comprehensive representation of contextual information compared to the input word embeddings and also compared to the the output of a BiLSTM encoder

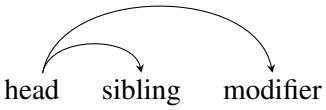


Figure 5: Sibling structure

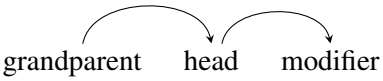


Figure 6: Grandchild structure

of a StackPtr.

4.3 Stack Pointer Network

Stack Pointer Network (StackPtr) (Ma et al., 2018) is a transition-based structure but it still performs a global optimization over the potential dependency parse trees of a sentence. StackPtr is based on a pointer network (PTR-NET) (Vinyals et al., 2015) but differently, a StackPtr has a stack to store the order of head words in trees. In each step, an arc is built from a child to the head word at the top of the stack based on the attention scores obtained from a pointer network.

We use a Stack Pointer Network for decoding the sequence to infer the dependencies, where each word is encoded with a Transformer Network as mentioned in the previous section.

The transformer encoder outputs a hidden state vector s_i for the ith word in the sequence. The hidden state vector is summed with higher-order information similar to that of Ma et al. (2018). There are two types of higher-order information in the model: Sibling (two words that have the same parent) and grandparent/grandchild (parent of the word's parent and the child of the word's child). Figure 5 and Figure 6 shows an illustration of these high-order structures.

So, the input vector for the decoder is the sum of the state vector of the word on the top of the stack, its sibling and its grandparent:

$$\beta_i = s_h + s_s + s_g \tag{2}$$

In the decoder part, an LSTM gathers all of the contextual and higher-order information about the word at the top of stack. Normally, in the pointer networks, at each time step t, the decoder receives the input from the last step and outputs decoder hidden state h_t. Therefore, an attention score is obtained as follows:

$$e_i^t = score(h_t, s_i) \tag{3}$$

where e^t is the output of the scoring function, s_i is the encoder hidden state and h_t is the decoder hidden state at time step t. After calculating the score for each possible output in the Biaffine attention mechanism, the final prediction is performed as follows with a softmax function to convert it into a probability distribution:

$$a^t = softmax(e^t) \tag{4}$$

where a^t is the output probability vector for each possible child word and e^t is the output vector of the scoring function.

In our model, scoring function is adopted from Deep Biaffine attention mechanism (Dozat and Manning, 2016):

$$e_i^t = h_t^T W s_i + U^t h_t + V^t s_i + b \tag{5}$$

where W is the weight matrix, U and V are the weight vectors and b is the bias.

Additionally, before the scoring function, an MLP is applied to the output of decoder, as proposed by Dozat and Manning (2016) to reduce the dimensionality.

As for the dependency labels, we also use another MLP to reduce the dimensionlity and then apply deep biaffine to score the possible labels for the word at the top of the stack.

4.4 Learning

We use cross-entropy loss for training the model similar to StackPtr. The probability of a parse tree y for a given sentence x under the parameter set θ is $P_\theta(y|x)$ and estimated as follows:

$$P_\theta(y|x) = \prod_{i=1}^{k} P_\theta(p_i|p_{<i}, x) \tag{6}$$

$$= \prod_{i=1}^{k} \prod_{j=1}^{l_i} P_\theta(c_{i,j}|c_{i,<j}, p_{<i}, x) \tag{7}$$

$p_{<i}$ denotes the preceding paths that have already been generated, $c_{i,j}$ represents the j^{th} word in the path p_i and $c_{i,<j}$ denotes all the proceeding words on the path p_i. Here, a path consists of a sequence of words from the root to the leaf.

The model learns the arcs and labels in the dependency tree simultaneously.

5 Experiments & Results

5.1 Datasets

We ran experiments on both Turkish and English. We used Penn Treebank (PTB) (Marcus et al., 1993) for English and IMST dataset (Sulubacak et al., 2016) in Universal Dependencies for Turkish.

As for the word embeddings, we used pre-trained Glove embeddings (Pennington et al., 2014) on Wikipedia and pre-trained Polyglot embeddings (Al-Rfou' et al., 2013) on Wikipedia for both Turkish and English.

5.2 Evaluation Metrics

For the evaluation, we used two different evaluation metrics: UAS and LAS, which are the standard metrics for dependency parsing.

UAS is a metric that is used to calculate the accuracy of predicting words' heads. In other words, it is the ratio of the number of correctly predicted heads to the total number of words in the dataset:

$$\frac{UAS = \#ofcorrectheads}{\#ofwords} \quad (8)$$

LAS is another metric for dependency parsing that measures the correctness of both heads and labels. In other words, it is the ratio of correctly predicted heads **and** labels to the total number of words in the dataset:

$$\frac{LAS = \#ofcorrecthead, labelpair}{\#ofwords} \quad (9)$$

5.3 Hyperparameters

In our experiments, we use similar configurations with the baseline models: StackPtr model by Ma et al. (2018) and Self-Attention mechanism by Vaswani et al. (2017). Differently from the baseline models, for the self-attended encoder stack; we used 6 layers because this configuration performs better with the Polyglot embeddings for both English and Turkish as seen in Table 1 and Table 2 for English and Turkish respectively.

5.4 Results

The results obtained from IMST dataset (Sulubacak et al., 2016) in Turkish is given in Table 3, along with the results of other related work. OUr results compared to other related work show competitive performance for Turkish language. Our model gives an UAS score of 74.43% and LAS score of 64.26% with Glove embeddings, whereas an UAS score of 76.81% and LAS score of 67.95% are obtained with Polyglot embeddings. Therefore, using Polyglot embeddings gives far better results in Turkish. This could be due to the size of the train set used for the Polyglot embeddings.

Layer	UAS
1	86.24
2	88.56
4	92.40
6	**94.23**
8	93.13

Table 1: Accuracy for different number of encoder layers for PTB Dataset (Marcus et al., 1993)

Layer	UAS
1	69.89
2	71.48
4	74.51
6	**76.81**
8	75.32

Table 2: Accuracy for different number of encoder layers for Turkish IMST Dataset (Sulubacak et al., 2016)

The results obtained from Penn Treebank dataset (Marcus et al., 1993) in English is given in Table 4. Our results again show competitive performance compared to other related work for English. Similar to the Turkish results, our model performs better with Polyglot embeddings. While Glove gives 93.43% UAS and 91.98% LAS, Polyglot gives 94.23% UAS and 92.67% LAS.

5.5 Error Analysis

5.5.1 Sentence Length

The main aim in this study is to utilize Transformer Networks to resolve the long-term dependencies problem in dependency parsing. We analyzed the accuracy of our model in both short and longer sentences to see the impact of the Transformer Networks in our model compared to sequential StackPtr model that is based on LSTMs.

Table 7 gives the results for different lengths of sentences to show the impact of using Transformer Networks in long term depedencies. We compare our model with the original StackPtr (Ma et al., 2018) model, which is based on LSTMs. As the results show, our model performs far better for sentences with more than 20 words compared to the standard StackPtr model, with an improvement

Model	UAS	LAS
Our Model w/ Glove	74.43	64.26
Our Model w/ Polyglot	76.81	67.95
Nguyen and Verspoor (2018)	70.53	62.55
Kondratyuk and Straka (2019)	74.56	67.44
McDonald et al. (2006)	74.70	63.20
Dozat and Manning (2016)	77.46	68.02
Ma et al. (2018)	79.56	68.93
Ballesteros et al. (2015)	79.30	69.28

Table 3: Results for Turkish IMST Dataset (Sulubacak et al., 2016)

Model	UAS	LAS
Our Model w/ Glove	93.43	91.98
Our Model w/ Polyglot	94.23	92.67
Ballesteros et al. (2015)	91.63	89.44
Chen and Manning (2014)	91.8	89.6
Kiperwasser and Goldberg (2016)	93.1	91.0
Ballesteros et al. (2016)	93.56	91.42
Weiss et al. (2015)	94.26	92.41
Andor et al. (2016)	94.61	92.79
Ma and Hovy (2017)	94.88	92.98
Dozat and Manning (2016)	95.74	94.08
Ma et al. (2018)	95.87	94.19

Table 4: Results for English PTB Dataset (Marcus et al., 1993)

of UAS score with around 7%.

For less than 20 words, our model's accuracy is lower compared to longer sentences. It shows that our self-attention based model is not able to learn shorter sentences better than the BiLSTM based StackPtr model. However, we observed that decreasing the number of layers in our encoder stack gives a higher accuracy for shorter sentences. However, it decreases the overall accuracy for the entire dataset.

5.5.2 The Impact of Punctuation

We also analyzed the impact of using punctuation in the datasets during training. Analysis of Spitkovsky et al. (Spitkovsky et al., 2011) shows that the usage of lexicalized and punctuated sentences gives better results in dependency parsing. So, we ran our model with both punctuated and not-punctuated versions of both datasets in Turkish and English. Table 5 shows that punctuation affects the learning of the model for both languages

Dataset	w/ Punctuation	w/o Punctuation
PTB	94.23 (92.67)	93.47 (91.94)
IMST	76.81 (67.95)	71.96 (62.41)

Table 5: Accuracy (UAS (LAS)) with and without punctuation on IMST (Sulubacak et al., 2016) and PTB (Marcus et al., 1993) Datasets

Input Embeddings	UAS
Glove	63.24
Polyglot	65.76
Polyglot + PoS	70.48
Polyglot + CNN	73.81
Polyglot + PoS + CNN	76.81

Table 6: The impact of using word embeddings (Glove or Polyglot), PoS tag embeddings and character-based word embeddings for the Turkish IMST Dataset (Sulubacak et al., 2016)

and the results are comparably higher when the punctuation is also used in the datasets. The impact of using punctuation is even more for Turkish language and both UAS and LAS are around %5 higher compared to training on datasets without punctuation.

5.5.3 The Impact of Using Embeddings

We analyzed the effect of using various embeddings in the Transformer encoder. As mentioned before, we utilize word embeddings, PoS tag embeddings and char embeddings obtained from CNN in our model. Table 6 shows the impact of the embeddings on the accuracy of the model. As the results show, character-level encoding plays a crucial role in our model because it helps to mitigate the OOV problem during training. We obtained the highest scores when Polyglot word embeddings, PoS tag embeddings and character-based word embeddings are incorporated in training.

6 Conclusion & Future Work

Our experiments show that using Self-Attention mechanism increases parsing accuracy especially in longer sentences in Turkish. However, our parser requires more data to learn better for also shorter sentences. The results also show that using character level word embeddings along with word embeddings and PoS tag embeddings gives the highest accuracy for our model.

We obtained the highest scores when we include

Number of words in sentence	UAS - StackPtr	UAS - Self-Attended StackPtr
less than 10 words	93.23	86.47
between 10 and 20 words	88.96	81.63
more than 20 words	56.49	62.33

Table 7: Accuracies for different lengths of sentences in IMST Dataset in Turkish (Sulubacak et al., 2016)

6 layers in our encoder stack by using Polyglot embeddings. Our results also show that including punctuation in the dataset improves the accuracy substantially.

We leave integrating morpheme-level information in especially morphologically rich languages such as Turkish as future work.

References

Rami Al-Rfou', Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria. Association for Computational Linguistics.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany. Association for Computational Linguistics.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. 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 344–354, Beijing, China. Association for Computational Linguistics.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization.

Miguel Ballesteros and Bernd Bohnet. 2014. Automatic feature selection for agenda-based dependency parsing. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 794–805, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms.

Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A. Smith. 2016. Training with exploration improves a greedy stack-lstm parser.

Xavier Carreras and Michael Collins. 2009. Non-projective parsing for statistical machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 200–209, Singapore. Association for Computational Linguistics.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017. Improved neural machine translation with a syntax-aware encoder and decoder. pages 1936–1945.

Hao Cheng, Hao Fang, Xiaodong He, Jianfeng Gao, and Li Deng. 2016. Bi-directional attention with agreement for dependency parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2204–2214, Austin, Texas. Association for Computational Linguistics.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336, Austin, Texas. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc V. Le. 2018. Semi-supervised sequence modeling with cross-view training.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing.

Gülşen Eryiğit, Joakim Nivre, and Kemal Oflazer. 2008. Dependency parsing of turkish. *Computational Linguistics*, 34(3):357–389.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2019. Left-to-right dependency parsing with pointer networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 710–716, Minneapolis, Minnesota. Association for Computational Linguistics.

Jenny Rose Finkel and Christopher D. Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of Human Language Technologies: The*

2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pages 326–334, Boulder, Colorado. Association for Computational Linguistics.

Katrin Fundel-Clemens, Robert Küffner, and Ralf Zimmer. 2007. Relex - relation extraction using dependency parse trees. *Bioinformatics (Oxford, England)*, 23:365–71.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750, Los Angeles, California. Association for Computational Linguistics.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1923–1933, Copenhagen, Denmark. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Tao Ji, Yuanbin Wu, and Man Lan. 2019. Graph-based dependency parsing with graph neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2475–2485, Florence, Italy. Association for Computational Linguistics.

Zhanming Jie, Aldrian Obaja Muis, and Wei Lu. 2017. Efficient dependency-guided named entity recognition. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 3457–3465. AAAI Press.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Dan Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing universal dependencies universally.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Distilling an ensemble of greedy dependency parsers into one mst parser.

Ying Li, Zhenghua Li, Min Zhang, Rui Wang, Sheng Li, and Luo Si. 2019. Self-attentive biaffine dependency parsing. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 5067–5073. AAAI Press.

Zuchao Li, Jiaxun Cai, Shexia He, and Hai Zhao. 2018. Seq2seq dependency parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3203–3214, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Ji Ma, Jingbo Zhu, Tong Xiao, and Nan Yang. 2013. Easy-first POS tagging and dependency parsing with beam search. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 110–114, Sofia, Bulgaria. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2017. Neural probabilistic model for non-projective MST parsing. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 59–69, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1414, Melbourne, Australia. Association for Computational Linguistics.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank.

Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal Stanford dependencies: A cross-linguistic typology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*, pages 4585–4592, Reykjavik, Iceland. European Languages Resources Association (ELRA).

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 216–220, New York City. Association for Computational Linguistics.

Alireza Mohammadshahi and James Henderson. 2019. Graph-to-graph transformer for transition-based dependency parsing.

Dat Quoc Nguyen and Karin Verspoor. 2018. An improved neural network model for joint. *Proceedings of the*.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, pages 950–958, Columbus, Ohio. Association for Computational Linguistics.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *Transactions of the Association for Computational Linguistics*, 5.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Peng Qi and Christopher D. Manning. 2017. Arc-swift: A novel transition system for dependency parsing.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 751–759, Los Angeles, California. Association for Computational Linguistics.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2011. Punctuation: Making a point in unsupervised dependency parsing. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 19–28, Portland, Oregon, USA. Association for Computational Linguistics.

Umut Sulubacak, Memduh Gokirmak, Francis Tyers, Çağrı Çöltekin, Joakim Nivre, and Gülşen Eryiğit. 2016. Universal dependencies for Turkish. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3444–3454, Osaka, Japan. The COLING 2016 Organizing Committee.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. 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 1556–1566, Beijing, 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 you need.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. 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 323–333, Beijing, China. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 562–571, Honolulu, Hawaii. Association for Computational Linguistics.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215, Brussels, Belgium. Association for Computational Linguistics.

Junru Zhou and Hai Zhao. 2019. Head-driven phrase structure grammar parsing on Penn treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2396–2408, Florence, Italy. Association for Computational Linguistics.

Creation of Corpus and Analysis in Code-Mixed Kannada-English Social Media Data for POS Tagging

Appidi Abhinav Reddy, Vamshi Krishna Srirangam,
Suhas Darsi and Manish Shrivastava
Language Technologies Research Centre (LTRC)
Kohli Centre on Intelligent Systems(KCIS)
International Institute of Information Technology, Hyderabad, India.
(abhinav.appidi, v.srirangam, darsi.suhas)@research.iiit.ac.in
m.shrivastava@iiit.ac.in

Abstract

Part-of-Speech (POS) is one of the essential tasks for many Natural Language Processing (NLP) applications. There has been a significant amount of work done in POS tagging for resource-rich languages. POS tagging is an essential phase of text analysis in understanding the semantics and context of language. These tags are useful for higher-level tasks such as building parse trees, which can be used for Named Entity Recognition, Coreference resolution, Sentiment Analysis, and Question Answering. There has been work done on code-mixed social media corpus but not on POS tagging of Kannada-English code-mixed data. Here, we present Kannada-English code-mixed social media corpus annotated with corresponding POS tags. We also experimented with machine learning classification models CRF, Bi-LSTM, and Bi-LSTM-CRF models on our corpus.

1 Introduction

The advent of social media like Twitter, Facebook, and Reddit has accelerated the communication between people of all colors, nations, and languages. Though the platform exists, the barriers to communication still exist due to languages. Many researchers are trying to solve this through various methods.

India is a land of multiple languages and majority of people are multilingual and tend to mix words from different languages in written text and also in speech. This interchanging language method involves complex grammar and is commonly addressed by terms 'Code-mixing' and 'Code-switching' as described by Lipski (1978). Code-switching refers to the use of words or phrases from different languages within the same speech context, whereas Code-mixing refers to the use of words or phrases from different languages in the same sentence. We can understand the difference between code-mixing and code-switching from the positions of altered elements. Code-mixing refers to the intra-sentential modification of codes, whereas code-switching refers to the inter-sentential modification of codes.

1.1 Characteristic of Code-Mixed Kannada-English Data

As explained above mixing happens at phrase, word, syntactic and morphological level too. Following are few more examples :

1. **Morphological level:** The word 'cinemagalu' in Kannada, the root word 'cinema' is borrowed from English and 'galu' is a Kannada morphene that marks plurality.

2. **Phrase level:** This is a completely code-mixed sentence. For example, 'Kelsa bittu pitch reporter aagu olle future ide!' which means 'Leave your work and become pitch reporter, you have great future in that!'. Here the statement follows the structure of Kannada with English words embedded in it.

3. **Word level:** This is language mixing occuring at word level. A complete word from English language is taken into Kannada language. An example: 'Ee thara branch ideya' which means 'Is there a branch like this?'.

4. **Syntactic level:** There are occurrences in Kannada-English CM data where inter-sentential mixing takes place. For example, 'Born and brought up in bengaluru, Yaako nange mysoor thumba ista, mysoor alli kelsa sikdre ready to shift.'

While there are robust solutions currently to handle non-code-mixed data, the same is not true for code-mixed data. One of the keys to solving any

Proceedings of the 17th International Conference on Natural Language Processing, pages 101–107
Patna, India, December 18 - 21, 2020.

higher-level NLP tasks is to do POS tagging. While POS tagging on English is very mature at this point, POS tagging for code-mixed in low-resource languages is relatively uncommon. In this paper, we have tried to address this problem. Here, we present Kannada-English code-mixed social media corpus annotated with corresponding POS tags.

Due to unstructured, informal, and incomplete information available in the data, it complicates the task of Code-mixed Kannada-English. Following are the challenges associated with the corpus.

- **Ambiguous words**: A word in one language can have a different meaning in other languages. For example, the word 'Bali' in English, which is a place in Indonesia, also used in Kannada which means 'Near'.

- **Word-level Code-mixing**: In the word 'Kanglish', its a fusion of two words Kannada and English at word level. This is similar to language mixing at word-level.

- **Word Orders**: English and Indian languages follow different word orders. Indian languages follow Subect-Object-Verb format, whereas English language follows Subject-Verb-Object format.

- **Reduplication**: People tend to use a second word with first word, which does not have a meaning on its own. The second word when addressed together with the first word it becomes a multi word expression. For example 'postu geestu', 'desha gesha', 'man ban'.

- **Variable Lexical Representations**: Users on social media have preference for their own way of native words like for example 'hogilla' is a Kannada word and it can be written as 'hogila', 'hgilla' etc.

Here are an instance depicting Kannada-English code-mixed nature and its translation.

T1 : *"@Suharsh2512 oho, idyaavdo brilliant facility. Nanna phone alli sound barutte.. ondond sala baralla. Hyaage nodu..."*
Translation: *"@Suharsh212 Oho...this is some brilliant facility...in my phone there is sound..once there is no sound...see how it is "*

2 Background and Related Work

POS tagging is a crucial stage in the NLP pipeline (Cutting et al., 1992) and has been explored extensively by Toutanova et al. (2003). Gimpel et al. (2010) and Owoputi et al. (2013) worked on the POS tagging of social media data. POS tagging for English using Dynamic Feature Induction with an accuracy of 97.64% was done on the WallStreet journal data set by Choi (2016).

POS tagging work has been done on Indian monolingual languages. Earlier work in POS tagging for Indian languages was mainly based on rule-based approaches (Antony and Soman, 2011). Some works in POS tagger system in Hindi done by Singh et al. (2006) and in the Bengali language was done by Ekbal et al. (2009) and in Telugu by RamaSree and Kusuma Kumari (2007).

Not many works were done on the POS tagger on Code-mixed data. POS taggers have been trained on Hindi-English code-mixed posts generated on Facebook (Vyas et al., 2014; Sharma et al., 2016). Only one public dataset of English-Hindi code-mixed Twitter posts annotated for POS tags exists (Jamatia and Das, 2016). Some of the recent works in code-mixed includes POS on code-mixed Telugu-English by Nelakuditi et al. (2016) and in NER in Telugu-English code-mixed social media data by Srirangam et al. (2019).

There are not many works done on Kannada because of the scarcity of quality annotated data. Recent works in POS tagging on Kannada were experimented only with traditional ML techniques like HMM, CRF, or SVM (BR and Kumar, 2012; Antony and Soman, 2010). Todi et al. (2018) built a Kannada POS tagger using machine learning and neural network models.

There have been very few works done on Kannada-English code-mixed data. Lakshmi and Shambhavi (2017) presented an automatic language identification system for code-mixed Kannada-English Social media text. Shalini et al. (2018) worked on sentiment analysis for Code-Mixed Kannada-English Social Media Text.

To the best of our knowledge, the corpus created for this paper is the first ever Kannada-English code-mixed social media corpus with POS tags.

3 Corpus Creation and Annotation

This corpus consists of Kannada-English code mixed tweets scraped from Twitter for the past six years based on topics such as sports, trending

hashtags, politics, movies, events, and others not limited to a particular domain. The tweets were collected using twintproject[1]-an opensource twitter intelligence tool. We retrieved over 318,000 tweets using the mentioned tool. After extensive cleaning and pre-processing of tweets, we were left with 6468 code-mixed Kannada-English tweets. We have done extensive pre-processing of tweets and retrieved them in JSON format. This JSON formatted data includes metadata like URLs, usernames, retweets, tweet IDs, likes, full names, and others.

The following steps were followed during pre-processing :

- Removing useless, noisy tweets, i.e., tweets containing only hashtags and URLS.
- Tweets that were written in only English or only Kannada were removed too.
- Tweets that having a minimum of ten words and contain linguistic units from both English and Kannada are only considered.
- Tweet Tokenizer is used to do Tokenisation of tweets.

The corpus will be made available for public use as soon as possible. The following explains the mapping of the tokens with their respective tags.

3.1 Annotation: Parts of Speech

Since the paper focuses on two different languages Kannada and English, we follow the Universal POS proposed by Petrov et al. (2011), which covers POS tags across all languages. There are 17 tags in the Universal POS[2], which we are following such as adjectives(ADJ), adposition(ADP), adverb(ADV), auxiliary(AUX), coordinating conjunction(CCONJ), determiner(DET), interjection(INTJ), noun(NOUN), numeral(NUM), particle(PART), pronoun(PRON), proper noun(PROPN), punctuation(PUNCT), subordinating conjunction(SCONJ), symbol(SYM), verb(VERB), and other(X). These tags are used in the annotation of our corpus. 'X' tag in the Universal POS is used to denote typos, foreign words, unknown abbreviations, and others. We included punctuation symbols under the category 'PUNC'. Following is an example of an annotated tweet and its translation.

[1]https://github.com/twintproject/twint

[2]https://universaldependencies.org/u/pos/

Tag	Cohen Kappa	Tokens
ADJ	0.84	6209
ADP	0.85	7000
ADV	0.85	11765
AUX	0.92	2098
CCONJ	0.83	2252
DET	0.88	3334
INTJ	0.87	943
NOUN	0.91	44533
NUM	0.92	1220
PART	0.89	569
PRON	0.89	17549
PROPN	0.91	7411
PUNCT	0.90	15602
SCONJ	0.82	1713
SYM	0.83	617
VERB	0.81	32545
X	0.85	1381

Table 1: Inter Annotator Agreement.

T2 : *"Haha/INTJ ashtu/ADV idea/NOUN illade/ADV gowdru/NOUN bengaluru/NOUN north/NOUN bittu/VERB tumukur/NOUN hogilla/ADV"*
Translation: "Haha without having much idea gowda left bengaluru north and went to tumukur."

3.2 Inter-annotator Agreement

Two people who are with linguistic backgrounds, both proficient in Kannada and English, manually did the annotations of the POS tags. Inter Annotator Agreement (IAA) is used to validate the quality of the annotation between two annotation sets of 6468 tweets and 156761 tokens using Cohen's Kappa coefficient (Hallgren, 2012) (refer Table 1 for Score). The agreement is significantly high.

4 Corpus Statistics

We have collected more than 318,000 of tweets from Twitter using TwintProject. After extensive cleaning, we were left with 6468 code-mixed Kannada-English tweets, as part of annotation using sixteen POS tags along with 'X' tag for foreign words, we tagged 156761 tokens (refer Table 1). We made sure that all the words in the corpus are in Roman script. We used hashtags related to sports, trending hashtags, politics, movies, events, and others in collecting the corpus.

5 Experiments

We present the experiments using a combination of features and systems. To understand the effect of different parameters and features of the model, we performed several experiments. With some set of features at once and all at a time simultaneously, we performed experiments while changing the parameters of the model, like regularization parameters and algorithms of optimization like 'L2 regularization', 'Average Perceptron'and 'Passive Aggressive' for CRF, optimization algorithms and loss functions in LSTM. We used three-fold cross-validation for CRF. We used 'scikit-learn,' 'Tensorflow,' and 'Keras' libraries to implement the above algorithms.

5.1 Conditional Random Field (CRF)

CRFs are type of discriminative undirected probabilistic graphical model. In natural language processing, linear chain CRFs are popular, which implement sequential dependencies in predictions.[3] It is a supervised learning method and most often used for structured prediction tasks. In CRF, a set of feature functions are defined to extract features for each word in a sentence. It has applications in NER, POS tagging, among others. When it comes to POS tagging, it has been proven to be better than the tree-based models.

5.2 LSTM

Long Short Term Memory (LSTM) is a special kind of RNN architecture that is well suited for classification and making predictions based on time series data. LSTMs are capable of capturing only past information. In order to overcome this limitation Bidirectional LSTMs are proposed where two LSTM networks run in forward and backward directions capturing the context in either directions.

5.3 LSTM-CRF

The Bi-LSTM-CRF is a combination of bidirectional LSTM and CRF (Huang et al., 2015; Lample et al., 2016). The Bi-LSTM model can be combined with CRF to enhance recognition accuracy. This combined model of Bi-LSTM-CRF inherits the ability to learn past and future context features from the Bi-LSTM model and use sentence-level tags to predict possible tags using the CRF layer. Bi-LSTM-CRF has been proved to be a powerful model for sequence labeling tasks like POS tagging, shallow parsing, and NER.

5.4 Features

The features to our machine learning models consist of lexical, word-level and character features such as char N-Grams of size 2 and 3 in order to capture the information from emojis, mentions, suffixes in social media like '#,' '@,' numbers in the string, numbers, punctuation. Features from adjacent tokens are used as contextual features.

1. **Character N-Grams:** Character N-Grams are proven to be efficient in the task of classification of text and are language-independent (Majumder et al., 2002). They are helpful when there are misspellings in the text (Cavnar et al., 1994; Huffman, 1995; Lodhi et al., 2002). Group of chars can help in capturing the semantic information. Character N-Grams are especially helpful in cases like code mixed language where there is free use of words, which vary significantly from the standard Kannada-English words.

2. **Word N-Grams:** Bag of words has been a staple for languages other than English (Jahangir et al., 2012) in tasks like NER and POS. Thus, we use adjacent words as a feature vector to train our model as our word N-Grams. These are also called contextual features. We used Word N-Grams of size 3 in the paper.

3. **Common Symbols:** It is observed that currency symbols, brackets like '(,' '[,' etc. And other symbols are followed by numeric or some mention, are present in the corpus which direct to symbol tag under Universal POS. Hence, the presence of these symbols is a good indicator of the words before or after them for being a 'SYM' tag in POS tagging.

4. **Numbers in String:** In social media, we see people using alphanumeric characters, generally to save the typing effort, to showcase their style or shorten the message length. When observed in our corpus, words containing alphanumeric are generally tagged under 'NUM' tag.

5. **Mentions and Hashtags:** People use '@' mentions to refer to persons or organizations, they use '#' hashtags in order to make something notable or to make a topic trending.

[3] https://en.wikipedia.org/wiki/Conditional$_{r}andom_{f}ield$

Thus the presence of these two gives a reasonable probability for the word being a named entity which counts under proper nouns.

6. **Capitalization:** In social media, people tend to use capital letters to refer to the names of persons, organizations and persons; at times, they write the entire name in capitals (Von Däniken and Cieliebak, 2017) to give particular importance or to denote aggression. This gives rise to a couple of binary features. One feature is to indicate if the beginning letter of a word is capitalized, and the other is to indicate if the entire word is capitalized.

6 Results and Discussion

Table 2 shows CRF results with 'l2-sgd' (Stochastic Gradient Descent with L2 regularization) algorithm for 200 iterations. The c2 value in the CRF model refers to the 'L2 regression'. Experiments using the algorithms 'pa' (Passive-Aggressive) and 'ap' (Averaged Perceptron) resulted in similar F1-scores of 0.79. The table 3 shows results after removing each particular feature. Example prediction of our CRF model is shown under appendix section.

In both the experiments Bi-LSTM and Bi-LSTM-CRF, we experimented with the optimizer, activation functions, and the number of epochs. After several experiments, the best result we came through was using 'softmax' as activation function, 'rmsprop' as an optimizer and 'categorical cross-entropy' as our loss function. Table2 shows the results of BiLSTM on our corpus using thirty epochs, and also shows the results of Bi-LSTM-CRF on our corpus using twenty epochs, both with random initialization of embedding vectors. The training, validation, and testing for both experiments are 60%, 10%, and 30% of the total data, respectively. Bi-LSTM resulted in best F1-score of 0.80 and Bi-LSTM-CRF with best F1-score of 0.81.

Tag	CRF	Bi-LSTM	BiL-CRF
ADJ	0.58	0.52	0.58
ADP	0.75	0.73	0.78
ADV	0.75	0.79	0.72
AUX	0.99	1.00	0.99
CCONJ	0.99	0.31	0.99
DET	0.85	0.74	0.88
INTJ	0.97	0.93	0.87
NOUN	0.83	0.84	0.84
NUM	0.69	0.76	0.74
PART	1.00	0.99	1.00
PRON	0.67	0.60	0.63
PROPN	0.86	0.77	0.77
PUNCT	1.00	1.00	1.00
SCONJ	0.77	1.00	0.75
SYM	0.80	0.74	0.78
VERB	0.70	0.70	0.69
X	0.79	0.80	0.81
weighted avg	0.80	0.79	0.79

Table 2: Table shows F1-scores for CRF, Bi-LSTM and Bi-LSTM-CRF respectively.

Feature removed	Precision	Recall	F1
Char N-Grams	0.66	0.50	0.45
Word N-Grams	0.62	0.53	0.50
Common Symbols	0.66	0.55	0.52
Numbers in String	0.62	0.56	0.55
Mentions, Hashtags	0.60	0.56	0.54
Capitalization	0.59	0.55	0.53

Table 3: Feature(removed) Specific Results for CRF.

7 Conclusion and Future Work

Our Contributions are as follows:

1. Presented an annotated Kannada-English code-mixed corpus for POS, which is, to the best of our knowledge is the first ever corpus. The corpus will be made available online.

2. We have experimented with the machine learning models CRF, Bi-LSTM, and Bi-LSTM-CRF on our data, the F1-score for which is 0.79, 0.80, and 0.81 respectively, which looks good considering the amount of research done in this new area.

3. We are introducing and addressing Part-of-Speech of code-mixed Kannada-English data as a research problem.

For future work, the corpus can also be enriched by giving the NER tags for each token. The size of the corpus can be increased with more data. The problem can be adapted for POS tagging in multilingual code-mixed data.

References

PJ Antony and KP Soman. 2010. Kernel based part of speech tagger for kannada. In *2010 International Conference on Machine Learning and Cybernetics*, volume 4, pages 2139–2144. IEEE.

PJ Antony and KP Soman. 2011. Parts of speech tagging for indian languages: a literature survey. *International Journal of Computer Applications*, 34(8):0975–8887.

Shambhavi BR and Ramakanth Kumar. 2012. Kannada part-of-speech tagging with probabilistic classifiers. *international journal of computer applications*, 48(17):26–30.

William B Cavnar, John M Trenkle, et al. 1994. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*, volume 161175. Citeseer.

Jinho D Choi. 2016. Dynamic feature induction: The last gist to the state-of-the-art. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 271–281.

Douglass Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. 1992. A practical part-of-speech tagger. In *Third Conference on Applied Natural Language Processing*, pages 133–140.

Asif Ekbal, Md Hasanuzzaman, and Sivaji Bandyopadhyay. 2009. Voted approach for part of speech tagging in bengali. In *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation, Volume 1*, pages 120–129.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. 2010. Part-of-speech tagging for twitter: Annotation, features, and experiments. Technical report, Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science.

Kevin A Hallgren. 2012. Computing inter-rater reliability for observational data: an overview and tutorial. *Tutorials in quantitative methods for psychology*, 8(1):23.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Stephen Huffman. 1995. Acquaintance: Language-independent document categorization by N-grams. Technical report, DEPARTMENT OF DEFENSE FORT GEORGE G MEADE MD.

Faryal Jahangir, Waqas Anwar, Usama Ijaz Bajwa, and Xuan Wang. 2012. N-gram and gazetteer list based named entity recognition for Urdu: A scarce resourced language. In *Proceedings of the 10th Workshop on Asian Language Resources*, pages 95–104.

Anupam Jamatia and Amitava Das. 2016. Task report: Tool contest on pos tagging for code-mixed indian social media (facebook, twitter, and whatsapp) text@ icon 2016.". *Proceedings of ICON*.

BS Sowmya Lakshmi and BR Shambhavi. 2017. An automatic language identification system for code-mixed english-kannada social media text. In *2017 2nd International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS)*, pages 1–5. IEEE.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

John Lipski. 1978. Code-switching and the problem of bilingual competence. *Aspects of bilingualism*, 250:264.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2(Feb):419–444.

P Majumder, M Mitra, and BB Chaudhuri. 2002. N-gram: a language independent approach to IR and NLP. In *International conference on universal knowledge and language*.

Kovida Nelakuditi, Divya Sai Jitta, and Radhika Mamidi. 2016. Part-of-speech tagging for code mixed english-telugu social media data. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 332–342. Springer.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 380–390.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2011. A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*.

RJ RamaSree and P Kusuma Kumari. 2007. Combining pos taggers for improved accuracy to create telugu annotated texts for information retrieval. *Dept. of Telugu Studies, Tirupathi, India*.

K Shalini, HB Barathi Ganesh, M Anand Kumar, and KP Soman. 2018. Sentiment analysis for code-mixed indian social media text with distributed representation. In *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1126–1131. IEEE.

Arnav Sharma, Sakshi Gupta, Raveesh Motlani, Piyush Bansal, Manish Srivastava, Radhika Mamidi, and Dipti M Sharma. 2016. Shallow parsing pipeline for

hindi-english code-mixed social media text. *arXiv preprint arXiv:1604.03136*.

Smriti Singh, Kuhoo Gupta, Manish Shrivastava, and Pushpak Bhattacharyya. 2006. Morphological richness offsets resource demand–experiences in constructing a pos tagger for hindi. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 779–786.

Vamshi Krishna Srirangam, Appidi Abhinav Reddy, Vinay Singh, and Manish Shrivastava. 2019. Corpus creation and analysis for named entity recognition in telugu-english code-mixed social media data. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 183–189.

Ketan Kumar Todi, Pruthwik Mishra, and Dipti Misra Sharma. 2018. Building a kannada pos tagger using machine learning and neural network models. *arXiv preprint arXiv:1808.03175*.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 conference of the North American chapter of the association for computational linguistics on human language technology-volume 1*, pages 173–180. Association for Computational Linguistics.

Pius Von Däniken and Mark Cieliebak. 2017. Transfer learning and sentence level features for named entity recognition on tweets. In *3rd Workshop on Noisy User-generated Text (W-NUT), Copenhagen, 7 September 2017*, volume 3, pages 166–171. ACL.

Yogarshi Vyas, Spandana Gella, Jatin Sharma, Kalika Bali, and Monojit Choudhury. 2014. Pos tagging of english-hindi code-mixed social media content. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 974–979.

A Appendices

A.1 Example Prediction of CRF

Word	Truth	Predicted
Haha	INTJ	INTJ
ashtu	ADV	VERB
idea	NOUN	NOUN
illade	ADV	VERB
gowdru	NOUN	NOUN
bengaluru	NOUN	NOUN
north	NOUN	NOUN
bittu	VERB	NOUN
tumukur	NOUN	NOUN
hogilla	ADV	ADV

Identifying Complaints from Product Reviews in Low-resource Scenarios via Neural Machine Translation

Raghvendra P. Singh[‡], Rejwanul Haque[⋆], Mohammed Hasanuzzaman[†] and Andy Way
The ADAPT Centre
[‡]School of Computing, Dublin City University, Dublin, Ireland
[⋆]School of Computing, National College of Ireland, Dublin, Ireland
[†]School of Computing, Cork institute of Technology, Cork, Ireland
raghvendra.singh6@mail.dcu.ie
rejwanul.haque,mohammed.hasanuzzaman,andy.way@adaptcentre.ie

Abstract

Automatic recognition of customer complaints on products or services that they purchase can be crucial for the organisations, multinationals and online retailers since they can exploit this information to fulfil their customers' expectations including managing and resolving the complaints. Recently, researchers have applied supervised learning strategies to automatically identify users' complaints expressed in English on Twitter. The downside of these approaches is that they require labeled training data for learning, which is expensive to create. This poses a barrier for them being applied to low-resource languages and domains for which task-specific data is not available. Machine translation (MT) can be used as an alternative to the tools that require such task-specific data. In this work, we use state-of-the-art neural MT (NMT) models for translating Hindi reviews into English and investigate performance of the downstream classification task (complaints identification) on their English translations.

1 Introduction

Almost all online retailers allow users to freely express their opinions and thoughts on products via their websites and relevant social media platforms. Customers who intend to purchase a product may take purchasing decisions based on the reviews of the product. Accordingly, commercial and retail companies consider product reviews as an important source of information, and could exploit this information to build their marketing tools and strategy, and to resolve any issues in relation to the product. This could also benefit users with suggestions on the quality of the products or services that they want to purchase. As for the number of reviews of a product posted by the users, they could range from several hundreds to tens of thousands. E-commerce companies and online retailers want to identify complaints given the reviews of a product for their own benefit. Likewise, customers who want to buy a product or service may need such information while avoiding having to consult thousands of reviews about the product.

In this context, Gupta et al. (2014) identified the relationship between users' purchase intent from their social media forums such as Quora[1] and Yahoo! Answers,[2] and Wang et al. (2015) investigated the problem of identifying purchase intent with using a list of seed intent-indicators (e.g. 'want to'). Haque et al. (2019b) extend the work of Wang et al. (2015) while increasing the coverage of the purchase intent indicators with the distributed vector representation of words using the continuous skip-gram model (Mikolov et al., 2013).

Recently, Preotiuc-Pietro et al. (2019) automatically identified complaints from tweets posted by social media users and potential customers. In Singh et al. (2020), we conducted a similar study in an attempt to identify complaints from opinionated texts (reviews) about products posted in a low-resource language, Hindi, from the the websites of the retail giant Amazon India[3] and the popular social media platform YouTube.[4] For investigating this problem in Hindi (Singh et al., 2020), as in Gupta et al. (2014); Wang et al. (2015); Haque et al. (2019b); Preotiuc-Pietro et al. (2019), we had to manually create labeled training data[5]

[1] www.quora.com
[2] www.answers.yahoo.com
[3] https://www.amazon.in/
[4] https://www.YouTube.com/
[5] https://github.com/MrRaghav/

Proceedings of the 17th International Conference on Natural Language Processing, pages 108–116
Patna, India, December 18 - 21, 2020.

by employing a number of human annotators. The process of creating such a data set is not easy; it is not only time-consuming and laborious but also a very expensive task.

In this context, Tebbifakhr et al. (2019) investigated possibility of exploiting MT in a specific NLP task in a language for which dedicated tools are not available due to the scarcity of task-specific training data. As in Tebbifakhr et al. (2019), in this work, we considered Hindi, an under-resourced Indic language, and investigate whether MT can play a role in complaint identification and eliminate the requirement for complaint identification tools for Hindi, which require labeled data for training, which is expensive to create. Accordingly, we study the following two scenarios while considering reviews about a variety of products from the the websites of Amazon India and YouTube expressed in Hindi as the test examples in our experiments, namely performance of the classifiers (complaints identifiers) built for English on the English translations of the Hindi reviews by (i) our MT systems and (ii) human translators. Note that as a part of our investigation, we created a labeled training dataset of English reviews about products posted in Amazon, and detail the data creation process and statistics in Section 2.2.

Unlike Tebbifakhr et al. (2019) who focus on improving a downstream task (i.e. sentiment classification) by controlling translations of an MT system but at the expense of translation quality, we customise our neural MT systems using the standard and commonly-used data augmentation and terminology-aware domain adaptation techniques (Jooste et al., 2020; Haque et al., 2020c; Nayak et al., 2020b; Parthasarathy et al., 2020) so that the translations produced by the MT systems can retain source-side stylistics property and semantics as much as possible. In other words, in this study, we aim to observe the performance of the English classifiers (complaint identifiers) on the translations of the Hindi reviews by the baseline, adapted/customised neural MT systems, and human translators.

The remainder of the paper is organised as follows. In Section 2, we detail how we created training data for our experiments. Section 3 describes our MT system building and setups. In Section 4, we present our experimental methodology for complaint classification. Section 5 presents our evaluation results, with some discussion. Section 6 concludes and provides avenues for further work.

2 Dataset Creation

This section details the creation of training data that has been used in this task.

2.1 The Hindi Review test data

In attempt to create an evaluation test data of reviews, we first collected reviews written in Hindi posted online. The reviews were taken from two different sources: (i) websites of Amazon India, and (ii) YouTube. Amazon India has around 180 million listed products and YouTube has 265+ million active users. In order to collect the reviews from the Amazon India websites, we used the *amazon-reviews-scraper Python library*[6] which takes a product name as input and provides reviews about the product across the different languages. Similarly, in order to collect the reviews from YouTube, we used the *YouTube-comment-downloader Python library.*[7] This script provided us with reviews on the products across the different languages. In order to remove noise (e.g. HTML tags, special characters) from reviews, we applied a number cleaning scripts including a language identifier.[8]

Each of the collected clean reviews is manually tagged with a particular category, namely complaint or non-complaint. For this, we followed the annotation scheme described in Singh et al. (2020). A sample of annotated test set is presented in Table 1. The statistics about the test set reviews are shown in Table 2. We can see from Table 2 that the test set contains 400 examples, with 200 complaints and 200 non-complaints reviews. The numbers of positive and negative examples are equal because we wanted to use a balanced test set in our experiments. Note that the Hindi re-

Complaints-mining-from-Hindi-product-reviews

[6] https://github.com/philipperemy/amazon-reviews-scraper. Accessed on August 2020

[7] https://github.com/egbertbouman/YouTube-comment-downloader. Accessed on August 2020.

[8] https://pypi.org/project/pycld2/

	Review	Label
Hi:	वो ज़िन्दगी जो हम जीना चाहते हैं	0
En:	The life we want to live	
Hi:	पर फेस अनलॉक चल नहीं रहा	1
En:	But face unlock is not working	
Hi:	हिन्दी माध्यम के लिए एक वरदान	0
En:	A boon for Hindi medium	
Hi:	पृष्ठों की क्वालिटी व छपाई बहुत ही खराब हैं	1
En:	The quality and printing of pages are very poor	
Hi:	समान की डिलवरी ही नहीं हुई	1
En:	The product was not delivered	

Table 1: Sample Hindi reviews from test set and their manual English translations.

	count	words (HI)	words (EN)
Reviews	400	5,141	4,762
Complaints	200	2,932	2,738
Non-Complaints	200	2,209	2,024

Table 2: Statistics of the test set reviews.

views have been manually translated into English and the statistics about the English translations of the Hindi reviews are shown in the third column of Table 2. In addition to the sample Hindi reviews, Table 1 shows the corresponding English translations of the Hindi reviews.

2.2 The English Review data

As discussed above, for complaint identification in English we required labeled training data for building classifiers (complaint identifiers). Accordingly, we created labeled training data for English. For this, we followed the data creation and annotation methods described in Singh et al. (2020). First, we took English reviews from Amazon review dump.[9] We sampled reviews from four different categories, namely Books, Cell_Phones_and_Accessories, Electronics, and Movies_and_TV. The Hindi reviews which we collected from the websites of Amazon India and YouTube were mainly on books and electronic goods. This is the reason why we considered English reviews on those four (related) product categories. As for data cleaning and preprocessing, we adopted the same steps as applied for Hindi (cf. Section 2.1). Table 3 presents the statistics of the English dataset (the training and development sets).

Table 3: Statistics of the train and development sets (English reviews).

	Reviews	Words	Complaints
Train set	8,026	3,84,467	4,013
Dev. set	400	17,873	200

3 The Hindi-to-English MT Systems

Our MT systems are Transformer models (Vaswani et al., 2017) which were trained using the Marian-NMT toolkit.[10] The tokens of the training, evaluation and validation sets are segmented into sub-word units using Byte-Pair Encoding (BPE) (Sennrich et al., 2016), and BPE is applied individually on the source and target languages. From our experiences (Jooste et al., 2020; Haque et al., 2020b,c; Nayak et al., 2020b,a; Parthasarathy et al., 2020) in the participation in the recent shared translation tasks (Barrault et al., 2020; Mayhew et al., 2020; Nakazawa et al., 2020) involving low-resource language pairs and domains, we found that the following configuration usually leads to the best results in the low-resource translation settings: (i) the BPE vocabulary size: 6,000, (ii) the sizes of the encoder and decoder layers: 4 and 6, respectively, and (iii) learning-rate: 0.0003. As for the remaining hyperparameters, we followed the recommended best setup from Vaswani et al. (2017). The early stopping criterion is based on cross-entropy; however, the final NMT system is selected as per the highest BLEU score on the validation set. The beam size for search is set to 6. We make our final NMT model with ensembles of 8 models that are sampled from the training run.

For building our baseline models (forward

[9] https://jmcauley.ucsd.edu/data/amazon/

[10] https://github.com/marian-nmt/marian

and backward), we used the IIT Bombay English-Hindi parallel corpus[11] (Kunchukuttan et al., 2017) that is compiled from a variety of existing sources, e.g. OPUS[12] (Tiedemann, 2012). After applying standard cleaning procedures including applying a language identifier[13] we are left with just over 1.1 million parallel sentence pairs. As for Hindi and English monolingual sentences for forward-translation and back-translation, respectively, we sampled them from the AI4Bharat-IndicNLP Corpus (Kunchukuttan et al., 2020) and Amazon review dump (cf. Section 2.2), respectively. Table 4 presents the corpus statistics. As above (cf. Section 2.1), for our development set we used 385 reviews from Amazon India and YouTube, which were then manually translated into English (cf. last row of Table 4).

	sentences	words (EN)	words (HI)
Train	1,102,511	22.4M	23.4M
Monolingual			
English	6.86M	121.3M	–
Hindi	7.82M	–	142.9M
Dev. set	385	6,952	7,209

Table 4: The Corpus statistics.

We present the performance of our MT systems in terms of the automatic evaluation metric BLEU (Papineni et al., 2002). Additionally, we performed statistical significance tests using bootstrap resampling methods (Koehn, 2004). We obtained the BLEU scores of our MT systems on the test set, and the scores are reported in Table 5. The first row of Table 5 represents our baseline Hindi-to-English MT system. The English-to-Hindi MT system which has been used to translate the English monolingual sentences (reviews) into Hindi produced 20.52 BLEU points on the development set. The BLEU scores of the MT systems (Base+BT and Base+BT+FT) trained on training data that consists of both authentic and (target- or/and source-original) synthetic parallel data are shown in the next two rows of Table 5. As in Caswell et al. (2019), in order to let the NMT model know that the given source is synthetic, we tag the source sentences of the synthetic data with the extra tokens.

	BLEU	
	devset	test set
Base	25.92	23.03
Base+BT	30.84	26.51
Base+BT+FT	30.89	26.85
Base+BT+FT+DA	31.52	27.49

Table 5: The BLEU scores of the English-to-Hindi NMT systems.

We observed that the review texts generally contain terms or product names, and terminology translation is a challenging task in MT (Haque et al., 2019a, 2020a). In order to adapt our best MT system, Base+BT+FT, to the task, we adopted the terminology-aware on-the-fly adaption method Jooste et al. (2020); Haque et al. (2020c); Nayak et al. (2020b); Parthasarathy et al. (2020), and mine those sentences from large monolingual datasets that could be beneficial for fine-tuning the original NMT model. As in Jooste et al. (2020); Haque et al. (2020c); Nayak et al. (2020b); Parthasarathy et al. (2020), we first identified terms in the review test set (cf. Table 2) to be translated,[14] and given the list of extracted terms, Hindi sentences which were mined from large monolingual data are similar in style to the test set sentences. We mined Hindi sentences (a total of 129,800 sentences) from a large monolingual corpus given the list of terms (a total of 2,953 terms) appearing in the test set. Then, a source-original synthetic corpus was created by translating these mined Hindi sentences into English using the best MT system, Base+BT+FT. The monolingual corpus that we used for this purpose contains 62,679,936 sentences from the AI4Bharat-IndicNLP Corpus. Additionally, we mined 36,397 sentences from the source side of the parallel training corpus and took their target counterparts, which gives us an authentic parallel corpus for adaptation. Finally, Base+BT+FT was fine-tuned on the resultant training corpus (166,197 training instances which contains 129,800 synthetic and 36,397 authentic sentence-pairs). As for translating the development set sentences, we fol-

[11] http://www.cfilt.iitb.ac.in/iitb_parallel/
[12] http://opus.lingfil.uu.se/
[13] https://pypi.org/project/pycld2/
[14] We followed Haque et al. (2014, 2018) in order to automatically identify terms in the in-domain texts.

lowed the same strategy.

The BLEU scores of the adapted MT system (Base+BT+FT+DA) on the test set are shown in the last row of Table 5. When we compare the original MT system with the adapted MT system, we see that the adapted version of Base+BT+FT, Base+BT+FT+DA, produces a 0.64 BLEU point (corresponding to 2.38% relative) improvement over Base+BT+FT, and the improvement is statistically significant.

4 The Complaint Identification Models

4.1 LSTM Network

Nowadays, recurrent neural networks (RNN), in particular long-short term memory (LSTM) (Hochreiter and Schmidhuber, 1997) hidden units, have proven to be an effective model for many classification tasks in NLP, e.g. sentiment analysis (Wang et al., 2016), text classification (Joulin et al., 2016; Zhou et al., 2016). RNN is an extension of the feed-forward neural network (NN), which has the gradient vanishing or exploding problems. LSTM deals with the gradient vanishing and exploding problems of RNN. An RNN composed of LSTM hidden units is often called an LSTM network. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. More formally, each cell in LSTM can be computed as follows:

$$X = \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} \quad (1)$$

$$f_t = \sigma(W_f \cdot X + b_f) \quad (2)$$

$$i_t = \sigma(W_i \cdot X + b_i) \quad (3)$$

$$o_t = \sigma(W_o \cdot X + b_o) \quad (4)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh\ (W_c \cdot X + b_c) \quad (5)$$

$$h_t = o_t \odot tanh\ (c_t) \quad (6)$$

where $W_i, W_f, W_o \in \mathbb{R}^{d \times 2d}$ are the weighted matrices and $b_i, b_f, b_o \in \mathbb{R}^d$ are the biases of LSTM, which need to be learned during training, parameterising the transformations of the input, forget and output gates, respectively. σ is the sigmoid function, and $\odot$ stands for element-wise multiplication. x_t includes the inputs of LSTM cell unit. The vector of hidden layer is h_t. The final hidden vector h_N represents the whole input review, which is passed to the *softmax* layer after linearising it into a vector whose length is equal to the number of class labels. In our work, the set of class labels includes complaint and non-complaint categories.

4.2 Classical Supervised Classification Models

Furthermore, we compare the LSTM network with classical supervised classification models. We employ the following classical supervised classification techniques in our experiments:

- Logistic Regression (LR)
- Decision Tree (DT)
- Random Forest (RF)
- Naïve Bayes (NB)
- Support Vector Machine (SVM)

These classical learning models (LR, DT, RF, NB and SVM) can be viewed as the baseline in this task. Thus, we obtain a comparative overview on the performance of different supervised classification models including the LSTM network.

4.3 Training Setup

In order to build LR, DT, RF and NB classification models, we use the well-known scikit-learn machine learning library,[15] and performed all the experiments with default parameters set by scikit-learn. As for the representation space, each review was represented as a vector of word unigrams weighted by their frequency in the reviews.

For the classifiers based on the neural networks, we use a 300-Dimensional word embeddings from *fastText*. We use the *sigmoid* activation function with the Adam optimizer (Kingma and Ba, 2014) and binary cross entropy loss function. The size of the input layer of the NN is 300. We employ layer normalisation (Ba et al., 2016) in the model. Dropout

[15] https://scikit-learn.org/stable/

(Gal and Ghahramani, 2016) between layers is set to 0.10. The size of the embedding and hidden layers are 300. The models are trained with learning-rate set to 0.0003 and reshuffling the training examples for each epoch.

5 Results and Discussion

We evaluate the performance our classifiers on the gold-standard test set (cf. Table 2) and report the evaluation results in this section. In order to measure a classifier's accuracy on the test set, we use three widely-used evaluation metrics: precision, recall and F_1 measures. The results obtained are reported in Table 6. The first five columns of Table 6 represent our baseline classifiers (i.e. the classical supervised classification models). We see from the table that these classifiers perform moderately to excellently and LR is the best-performing method among them (LR: a 70.35 F_1 score)) when tested on the translations by our best MT system (Base+BT+FT+DA). This classifier (LR) produces an F_1 score of 71.47 on the gold-standard test set (i.e. translations of the Hindi reviews by translators).

Table 6: Performance of the classifiers on the evaluation test set.

	NB	LR	DT	SVM	RF	LSTM
Base						
P	57.64	66.98	62.82	66.82	68.78	75.76
R	41.50	72.00	73.5	69.50	70.50	75.00
F_1	48.26	69.4	67.74	68.13	69.63	75.38
Base+BT						
P	58.78	71.28	61.50	70.00	69.85	77.44
R	43.50	69.50	69.50	63.00	69.50	75.50
F_1	50.00	70.38	65.26	66.32	69.67	76.46
Base+BT+FT						
P	59.49	70.71	64.38	68.98	70.62	76.12
R	47.00	70.00	70.5	64.50	68.5)	76.5
F_1	52.51	70.35	67.30	66.67	69.54	76.31
Base+BT+FT+DA						
P	58.17	70.71	65.33	68.56	70.47	77.16
R	44.50	70.00	73.50	66.50	68.00	76.00
F_1	50.43	**70.35**	69.18	67.51	69.21	**76.58**
Manual (Upper Bound)						
P	58.22	73.55	63.55	71.82	71.42	80.10
R	42.50	69.50	68.00	65.00	67.50	78.50
F_1	49.13	**71.47**	65.70	68.24	69.41	**79.29**
Hindi classifiers on Hindi reviews						
P	59.09	74.14	77.77	72.16	84.51	74.71
R	65.00	76.00	31.50	70.00	30.00	65.00
F_1	61.91	**75.06**	44.84	71.07	44.28	69.52

As for our NN-based classifier, the LSTM network trained on fastText embeddings performed excellently as we see from Table 6, where it obtains an excellent F_1 score (76.58 F_1) on the test set of translations of the Hindi reviews by Base+BT+FT+DA. It obtains an F_1 score of 79.29 on the test set of translations of the Hindi reviews by human translators.

For comparison, we also measured the performance of the Hindi classifiers that were built on the Hindi training data released by Singh et al. (2020) on the original reviews (i.e. the Hindi-side of the test set; cf. Table 2), and the results are shown in the last rows of Table 6. We see from Table 6 that in this case, the best-performing Hindi classifier is LR and it produces an F_1 of 75.06, which is 1.52 F_1 points lower than that produced by the best-performing English classifier on the translations by our best MT system.

We clearly see from the scores presented in Table 6 that the performance of the English classifiers on the translations produced by our customised MT system (Base+BT+FT+DA) is comparable to that of the Hindi classifiers on the original Hindi reviews. Thus, we can say that MT when customised or trained to translate texts of specific styles (e.g. reviews about a variety of products) can act as an alternative to the tools that rely on task-specific (in this work, complaint identification) training data which is expensive to prepare.

6 Conclusion

In this paper, we presented a strategy in which MT can be used to eliminate the requirement for expensive task-specific data creation for low-resource languages or domains. We investigated our strategy on complaint identification from reviews about products posted in Hindi. We used state-of-the-art NMT models for translating the Hindi reviews into English, and investigate the performance of the English complaint identifiers on the translations of the Hindi reviews by the Hindi-to-English MT systems. For comparison, we tested the performance of the English classifiers on two setups: (i) English translations of the Hindi reviews by the MT systems, and (ii) gold-standard English translations of the Hindi reviews. We also compared the performance of the Hindi

classifiers built on a publicly available Hindi review training data set on the original Hindi reviews (i.e. the Hindi-side of the review test set).

In our experiments, we also aimed at preserving source-language stylistic properties and semantics in translation. For this, we applied standard and commonly-used data augmentation techniques and terminology-aware domain adaptation method (Jooste et al., 2020; Haque et al., 2020c; Nayak et al., 2020b; Parthasarathy et al., 2020) for building our Hindi-to-English NMT systems, and used task-specific target-language monolingual data. These strategies were found to be effective in this task. We demonstrated that the NMT systems when customised or trained to translate texts of specific styles (e.g. user-generated content or reviews) can act as an alternative to those tools that require task-specific (i.e. complaint identification) training data which are expensive to create.

We believe that this work would bring additional value to the social media analytics research and practice given the fact that many task-specific data are available in English only and does not exist in many low-resource and even some high-resource languages.

In future, we intend to test our method on different low-resource and high-resource non-English languages. We also plan to investigate this method on different NLP tasks.

Acknowledgments

This research has been supported by the ADAPT Centre for Digital Content Technology which is funded under the Science Foundation Ireland (SFI) Research Centres Programme (Grant No. 13/RC/2106) and is co-funded under the European Regional Development Fund.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Loïc Barrault, Magdalena Biesialska, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Matthias Huck, Eric Joanis, Tom Kocmi, Philipp Koehn, Chi-kiu Lo, Nikola Ljubešić, Christof Monz, Makoto Morishita, Masaaki Nagata, Toshiaki Nakazawa, Santanu Pal, Matt Post, and Marcos Zampieri. 2020. Findings of the 2020 conference on machine translation (wmt20). In *Proceedings of the Fifth Conference on Machine Translation*, pages 1–54, Online. Association for Computational Linguistics.

Isaac Caswell, Ciprian Chelba, and David Grangier. 2019. Tagged back-translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 53–63, Florence, Italy. Association for Computational Linguistics.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. *CoRR*, abs/1512.05287.

Vineet Gupta, Devesh Varshney, Harsh Jhamtani, Deepam Kedia, and Shweta Karwa. 2014. Identifying purchase intent from social posts. In *Proceedings of the Eighth International AAAI Conference on Weblogs and Social Media*, pages 180–186, Ann Arbor, Michigan.

Rejwanul Haque, Mohammed Hasanuzzaman, and Andy Way. 2019a. Investigating terminology translation in statistical and neural machine translation: A case study on English-to-Hindi and Hindi-to-English. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 437–446, Varna, Bulgaria.

Rejwanul Haque, Mohammed Hasanuzzaman, and Andy Way. 2020a. Analysing terminology translation errors in statistical and neural machine translation. *Machine Translation (in press)*, 34(2):149–195.

Rejwanul Haque, Yasmin Moslem, and Andy Way. 2020b. The ADAPT system description for the STAPLE 2020 English-to-Portuguese translation task. In *Proceedings of the Fourth Workshop on Neural Generation and Translation*, pages 144–152, Online. Association for Computational Linguistics.

Rejwanul Haque, Yasmin Moslem, and Andy Way. 2020c. Terminology-aware sentence mining for nmt domain adaptation: Adapt's submission to the adap-mt 2020 english-to-hindi ai translation shared task. In *Proceedings of the Workshop on Low Resource Domain Adaptation for Indic Machine Translation (Adap-MT 2020)*, Patna, India (to appear).

Rejwanul Haque, Sergio Penkale, and Andy Way. 2014. Bilingual termbank creation via log-likelihood comparison and phrase-based statistical machine translation. In *Proceedings of the 4th International Workshop on Computational*

Terminology (Computerm), pages 42–51, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Rejwanul Haque, Sergio Penkale, and Andy Way. 2018. TermFinder: log-likelihood comparison and phrase-based statistical machine translation models for bilingual terminology extraction. *Language Resources and Evaluation*, 52(2):365–400.

Rejwanul Haque, Arvind Ramadurai, Mohammed Hasanuzzaman, and Andy Way. 2019b. Mining purchase intent in twitter. *Computación y Sistemas*, 23(3).

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Wandri Jooste, Rejwanul Haque, and Andy Way. 2020. The ADAPT Centre's neural MT systems for the WAT 2020 document-level translation task. In *Proceedings of the the 7th Workshop on Asian Translation (WAT 2020), AACL-IJCNLP 2020*, pages 142–146, Suzhou, China.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Anoop Kunchukuttan, Divyanshu Kakwani, Satish Golla, Avik Bhattacharyya, Mitesh M Khapra, Pratyush Kumar, et al. 2020. Ai4bharat-indicnlp corpus: Monolingual corpora and word embeddings for indic languages. *arXiv preprint arXiv:2005.00085*.

Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. 2017. The IIT Bombay English–Hindi parallel corpus. *CoRR*, 1710.02855.

Stephen Mayhew, Klinton Bicknell, Chris Brust, Bill McDowell, Will Monroe, and Burr Settles. 2020. Simultaneous translation and paraphrase for language education. In *Proceedings of the Fourth Workshop on Neural Generation and Translation*, pages 232–243, Online. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Toshiaki Nakazawa, Hideki Nakayama, Chenchen Ding, Raj Dabre, Hideya Mino, Isao Goto, Win Pa Pa, Anoop Kunchukuttan, Shantipriya Parida, Ondřej Bojar, and Sadao Kurohashi. 2020. Overview of the 7th workshop on Asian translation. In *Proceedings of the 7th Workshop on Asian Translation*, Suzhou, China. Association for Computational Linguistics.

Prashanth Nayak, Rejwanul Haque, and Andy Way. 2020a. The ADAPT Centre's participation in WAT 2020 english-to-odia translation task. In *Proceedings of the the 7th Workshop on Asian Translation (WAT 2020), AACL-IJCNLP 2020*, pages 114–117, Suzhou, China.

Prashanth Nayak, Rejwanul Haque, and Andy Way. 2020b. The ADAPT's submissions to the WMT20 biomedical translation task. In *Proceedings of the Fifth Conference on Machine Translation (Shared Task Papers (Biomedical)*, pages 839–-846, Online Conference.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Venkatesh Balavadhani Parthasarathy, Akshai Ramesh, Rejwanul Haque, and Andy Way. 2020. The ADAPT system description for the WMT20 news translation task. In *Proceedings of the Fifth Conference on Machine Translation*, pages 261-–267, Online Conference.

Daniel Preotiuc-Pietro, Mihaela Gaman, and Nikolaos Aletras. 2019. Automatically identifying complaints in social media. *arXiv preprint arXiv:1906.03890*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Raghvendra Pratap Singh, Rejwanul Haque, Mohammed Hasanuzzaman, and Andy Way. 2020. Identifying complaints from product reviews: A case study on hindi. In *Proceedings of the Irish Conference on Artificial Intelligence and Cognitive Science (AICS 2020)*, pages 217–228, Dublin, Ireland.

Amirhossein Tebbifakhr, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2019. Machine translation for machines: the sentiment classification use case. In *Proceedings of the 2019 Conference*

on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 1368–1374, Hong Kong, China.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC'2012)*, pages 2214–2218, Istanbul, Turkey.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Jinpeng Wang, Gao Cong, Wayne Xin Zhao, and Xiaoming Li. 2015. Mining user intents in twitter: A semi-supervised approach to inferring intent categories for tweets. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 318–324, Austin, TX.

Yequan Wang, Minlie Huang, Li Zhao, et al. 2016. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615, Austin, TX.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. *arXiv preprint arXiv:1611.06639*.

Generative Adversarial Networks for Annotated Data Augmentation in Data Sparse NLU

Olga Golovneva
Amazon / Cambridge, MA
olggol@amazon.com

Charith Peris
Amazon / Cambridge, MA
perisc@amazon.com

Abstract

Data sparsity is one of the key challenges associated with model development in Natural Language Understanding (NLU) for conversational agents. The challenge is made more complex by the demand for high quality annotated utterances commonly required for supervised learning, usually resulting in weeks of manual labor and high cost. In this paper, we present our results on boosting NLU model performance through training data augmentation using a sequential generative adversarial network (GAN). We explore data generation in the context of two tasks, the bootstrapping of a new language and the handling of low resource features. For both tasks we explore three sequential GAN architectures, one with a token-level reward function, another with our own implementation of a token-level Monte Carlo rollout reward, and a third with sentence-level reward. We evaluate the performance of these feedback models across several sampling methodologies and compare our results to upsampling the original data to the same scale. We further improve the GAN model performance through the transfer learning of the pretrained embeddings. Our experiments reveal synthetic data generated using the sequential generative adversarial network provides significant performance boosts across multiple metrics and can be a major benefit to the NLU tasks.

1 Introduction

Over recent years, various task-oriented conversational agents, such as Amazon Alexa, Apple's Siri, Google Assistant, and Microsoft's Cortana, have become more popular in people's everyday life and are expected to be highly intelligent. For the NLU component, this means that we expect models to perform recognition of the actions and entities within a user's request with high accuracy. When first training an NLU model on a new language (a process referred to as bootstrapping a new language), there is a strong requirement for high quality annotated data that would support the most common user requests across a range of domains. As the modeling space expands to support new features and additional languages, NLU models are regularly re-trained on updated data sets to ensure support for these new functions. The major bottleneck in both of these processes is the labor and cost associated with collecting and annotating new training utterances for every new feature or language.

Recent advances in machine learning methods, including the use of techniques such as transfer learning (Lu et al., 2015) and active learning (Settles, 2009), can lead to more efficient data usage by NLU models and therefore decrease the need for annotated training data. Additionally, data augmentation models are being widely explored. The advantage of data augmentation is that once synthetic data is generated, it can be ingested into subsequent models without additional effort, allowing for faster experimentation.

NLU models in dialog systems can perform a variety of tasks (Ram et al., 2018; Gao et al., 2018). In this study, we will focus on three of them: **Domain classification (DC)** – identify the domain that the user request belongs to (music, reminders, alarm, etc.), **Intent classification (IC)** – extract actions requested by users (play music, find a restaurant, set an alarm, etc.), and **Named Entity Recognition (NER)** – identify and extract entities (names, values, dates, locations, etc.) from user requests.

For each utterance we expect our NLU model to output a domain, intent, and set of extracted entities with corresponding tags. For example, if a user requests "*play Bohemian Rhapsody by Queen*", we expect the NLU model to return {**domain**: *music*, **intent**: *play_song*, **named_entities**: [(bohemian

Proceedings of the 17th International Conference on Natural Language Processing, pages 117–126
Patna, India, December 18 - 21, 2020.

rhapsody, *song_name*), (queen, *artist_name*)]}. We call this output *annotation*, and the utterance along with annotation is called an *annotated utterance*. Named entities with corresponding labels are called *slots*.

For our NLU model to perform well on real-time user requests, we need to train it on a large dataset of diverse annotated utterances. However, there could be some areas of functionality where large datasets for training are not available. To boost model performance in situations where training data is limited, we use synthetic data generated from a small set of unique utterances that cover the basic functionality of the user experience, called *Golden utterances*. We leverage a Sequence Generative Adversarial Networks (SeqGAN) introduced by Yu et al. (2017) to generate new utterances from this "seed" set, and use these generated utterances to augment training data and evaluate the performance of the classification and recognition tasks. We also investigate how the metrics that we use to evaluate the quality of the generated synthetic data links to the performance boost in the underlying tasks.

2 Related work

NLU model boosting through training data augmentation has been an active area of research over the last few years, with more sophisticated techniques and models being developed. Some of these techniques include data resampling, the use of Variational Autoencoders (VAEs) and GANs. Xie et al. (2017) generalize resampling methods by proposing noising schemes that are designed to smooth input data by randomly changing the word tokens in a sentence. First described by Kingma and Welling (2013), VAEs learn distributed representations of latent variables, and decode random samples to generate data that have similar characteristics to those that the network was trained on. GAN model proposed by Goodfellow et al. (2014) includes two competing neural networks: a generator that creates fake data, and a discriminator that is trained to distinguish between fake and real data. The generator is trained on the results of its success in fooling the discriminator and this contest results in synthetic data that is progressively more similar to real data.

Synthetic data have shown to be useful for IC model boosting. For example, Malandrakis et al. (2019) explored a set of encoder-decoder models and proposed the use of conditional VAEs (CVAEs) to generate phrase templates, called carrier phrases. Authors used CVAEs to control the domain, intent, and slot types to generate desirable outputs that resulted in a higher F1 score on the intent classification task.

Kumar et al. (2019) focused on a few-shot IC problem where new categories with limited training data are introduced into an existing system with mature categories. They compared different techniques that were designed to augment training data, including upsampling, random perturbation, extrapolation, CVAEs, and delta-encoders, and combined feature space augmentation with popular BERT pre-training (Devlin et al., 2019) to provide better performance.

The use of GANs has been previously explored for text data augmentation in language modeling (Kusner and Hernández-Lobato, 2016; Yu et al., 2017; Che et al., 2017; Guo et al., 2018; Hu et al., 2017; Li et al., 2017; Lin et al., 2017; Zhang et al., 2017; Fedus et al., 2018) and sentiment classification (Gupta, 2019). However, discrete text sequence generation brings about several challenges: first, one needs to generate a set of discrete tokens from a random sample of real-valued continuous data, and second, GANs are designed to give feedback on entire sequences, whereas generators need guidance for each subsequent token. The SeqGAN model developed by Yu et al. (2017) attempts to resolve these issues by applying reinforcement algorithms for the GAN objective with a policy gradient that evaluates current state-action value using Monte Carlo (MC) search. In this work, we adopt a SeqGAN model to boost DC, IC, and NER tasks in NLU models that suffer from sparse data limitations.

3 Methods

3.1 Data

For our experiments, we used the English data[1] collected by Schuster et al. (2019). This data was consisted of three domains: *weather*, *alarm*, and *reminder*, and a total of 43000 utterances. It was collected in a three-step process: step 1 consisted of native English speakers producing utterances for each intent, step 2 consisted of two annotators labeling the intents and slots while any conflicts between these two annotators were resolved in step 3 by a third annotator. The data was processed

[1] https://fb.me/multilingual_task_oriented_data

further to match the format that was suitable for our models.

3.2 Models

Text data boosting in NLU was extensively used for classification tasks, so most previous research focused on generating sentences (Kumar et al., 2019), carrier phrases (Malandrakis et al., 2019), or embeddings (Guo et al., 2018). In our work we consider DC, IC, and NER problems, where both sequence and word tags are needed for model training. We leverage a GAN to synthesize training data as a sequence of intents and slots: $X = \{x_0, x_1, \ldots, x_n\}$, where n varies between a length of 1 and the maximum allowed utterance length. Each slot $x_i (i > 0)$ denotes the combination of the i^{th} word and its corresponding tag, and x_0 is a concatenation of the utterance domain and intent. For example, for the utterance "*play Bohemian Rhapsody by Queen*", the training sequence for text generation would be as follows: "*music/play_song play:none bohemian:song_name rhapsody:song_name by:none queen:artist_name*".

3.2.1 SeqGAN model

When applied to text data, traditional GANs have difficulty performing back-propagation due to the non-differentiable output of the generator model. The SeqGAN model addresses this issue by treating the generator as a reinforcement learning agent that optimizes the GAN objective. The discriminator itself is used within the reward function to evaluate output sequences and return feedback to guide the learning of the generative model. Traditional reward functions from classification models are also limited by the ability to only provide score/loss-based reward values for a complete sequence. The SeqGAN model enables evaluation of the action-value for an intermediate state of unfinished sequence for each initial state, s_0, by applying an MC search with a rollout policy to sample the unknown last tokens (Yu et al., 2017). MC search is a tree-search algorithm with a root node s_0. Each child node of the tree is drawn from a distribution parametrized by a stochastic parametrized policy. This policy can be any, for example current state of the generator G_θ.

In our experiments, we compare different ways to compute the reward function (Figure 1). First, we use an implementation where the MC tree search strategy is replaced with token-level reward produced by the discriminator (Xu et al., 2018; Hu et al., 2018). We use the output of the Long Short-Term Memory (LSTM)-based discriminator model D^L_ϕ, cross-entropy, as the reward. For a synthetic sentence $Y = \{y_0, y_1, ..., y_n\}, y_i \in \mathcal{Y}$, where $\mathcal{Y}$ is the vocabulary of candidate domain-intent and token-label pairs, the cross-entropy based reward for the i^{th} word is calculated as:

$$R(y_i) = -\log D^L_\phi(y_i|y_{<i}) \quad (1)$$

Next, we devise our own MC search-based method to produce a set of possible sequences to approximate the expectation of the token-level reward, rather than using a single evaluation of only one of the possible sequences. Using the current state of the generator model G_θ as a stochastic rollout policy, for each incomplete sequence $\{y_0, y_1, ..., y_k\}$ we use Monte-Carlo search to produce N complete sequences. We evaluate the token-level reward for each of these sequences:

$$\{y_{k+1}, ..., y_n\} \in MC^{G_\theta}(\{y_0, ..., y_k\}; N), \quad (2)$$

for $k < n$. To approximate the expected value of the reward function we average the token-level reward (1) calculated for the N MC rollouts:

$$R(y_i) = \frac{1}{(n+1)N} \sum_{k=0}^{n} \sum_{j=1}^{N} (-\log D^L_\phi(y_i|y_{<i}))_{j,k}, \quad (3)$$

where $(...)_{j,k}$ is the j_{th} MC rollout of the k_{th} sequence. Finally, we compare results with an expectation of a sentence-level reward calculated using MC tree rollout strategy for the convolutional neural network (CNN)-based discriminator feedback D^C_ϕ, that provides feedback on the full sentence (Yu et al., 2017). Intermediate token-level reward for current token y_i is approximated using MC rollouts as follows:

$$R_{MC}(y_i) = \frac{1}{N} \sum_{j=1}^{N} (D^C_\phi(Y))_j, i < n, \quad (4)$$

where the sampling of the missing $(n - i)$ tokens is governed by the equation (2). At each step i, generator reward is governed by the Monte-Carlo approximation (4), or discriminator feedback on the full sentence:

$$R(y_i) = \begin{cases} R_{MC}(y_i), i < n, \\ D^C_\phi(Y), i = n. \end{cases} \quad (5)$$

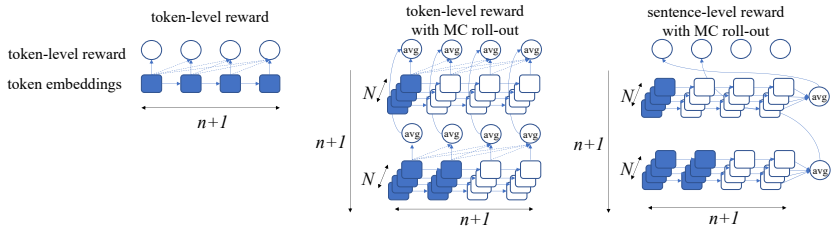


Figure 1: Schematic representation of the reward function produced by discriminator model for each of the $n+1$ input tokens and N Monte-Carlo rollouts *(avg = average)*. Tokens fixed at each iteration are represented by solid squares and tokens produced by the rollout policy are represented by blank squares. Sentence-level reward produces feedback on a complete sentence only, whereas token-level reward gives feedback on each token conditioned on previous set of tokens.

The generator is trained to maximize the reward from the discriminator model for the full sequence (Sutton et al., 2000):

$$J(\theta) = \mathbb{E}[R(Y|s_0,\theta)] = \sum_{y_0 \in \mathcal{Y}} G_\theta(y_0|s_0) Q^{G_\theta}_{D_\phi}(s_0, y_0) \quad (6)$$

where $Q^{G_\theta}_{D_\phi}(s_0, y_0)$ can be estimated by $R(y_0)$ using equations (1), (3) or (5).

The pre-training procedure for the generative decoder uses the maximum likelihood estimation metric, followed by the pre-training of the discriminative classifier on positive samples from the training data and negative samples produced by the generator. After pre-training, the generative and discriminative models are trained one at a time in the following loop: the last discriminator state is used to provide feedback for the generative model training, which then provides a new set of negative examples for the discriminator updates.

3.2.2 DC and IC/NER models

Our model architecture consists of two parts: a sentence classification model for the DC task, and a domain-specific joint model for the IC and NER tasks. The DC model is the same as the IC part of the joint IC-NER model, so we will describe only the latter in detail. The IC-NER model components are schematically shown in Figure 2, and are composed of the following:

- **Embedding**: concatenation of word embedding with 256 dimensions and character embedding with 16 dimensions trained on a 1-filter CNN with a *tanh* activation function and dropout.
- **Encoder**: 2-layer bidirectional LSTM with 384 dimensions in each hidden layer with dropout and layer normalization for token encoder, and a pooling stack for sequence encoder.
- **Decoders**: MLP classifier with 256 dimensions for IC task and CRF sequence labeler with 192 dimensions. For each block we also apply ELU activation function and dropout. All IC-NER models were trained for 500 epochs, and DC models for 100 epochs.

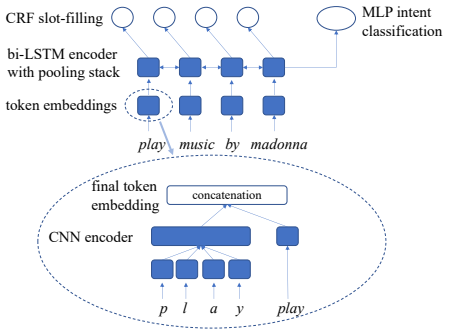


Figure 2: Multi-task model for IC and NER prediction.

3.3 Experimental setup

We build our experiments to mimic two major tasks that are of interest to us (Figure 3):

1. Bootstrapping of a new language
2. Handling low resource features

We manipulate our English data set to simulate modeling conditions in situations where training data is limited. For our tasks, we need to identify a set of utterances from within our data set that can be classified as Golden utterances, i.e., utterances that are found to be usually common

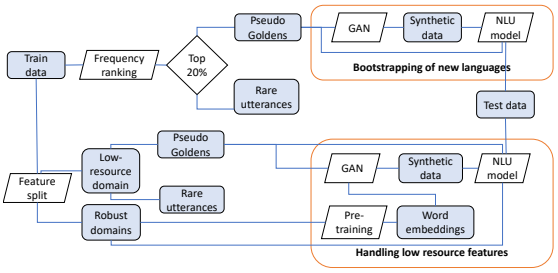


Figure 3: This schematic represents our data and experimental process. For our experiments we use annotated data collected by Schuster et al. (2019). Train and validation sets were combined together in a single train data set to extract pseudo Golden utterances. Pseudo Goldens were further used to train and evaluate the GAN model. The test set was untouched and was used to evaluate model performance in full scale. To simulate the low resource feature task, we used the full train data set for the two robust domains, and the pseudo Goldens to represent the third low-resource domain.

across languages and cover the basic functionality of the user experience. To select Goldens from our data set we use the following process. We sort the utterances in our training data by the utterance frequency within each domain-intent combination. We then select the top 20% of utterances within each domain-intent combination, unique them, and call them *pseudo Goldens*. The unique utterances within the remaining set are called *Rare* utterances. Utterance counts are provided in Appendix A.

3.3.1 Bootstrapping NLU models for new languages

When bootstrapping an NLU model to support a new language for a conversational artificial intelligence (AI), there is usually a very limited data available. One major data source is Goldens translated from another existing robust language model. In mimicking this task, we use our determined pseudo Goldens as input for both data generation and NLU models. Synthetic data generated with the GAN model is added to the set of pseudo Goldens and fed into our NLU model. The NLU models are tested using our fixed original test data set (Figure 3). Our baseline for comparison consists of an NLU model run on only the pseudo Goldens.

3.3.2 Handling low resource features

Another common challenge faced by conversational AI NLU models is sparse data. When a feature is new, or not common, we do not have enough data for the NLU model to generalize well on possible request variations. But, unlike the previous task, we typically do have a large amount of data collected in the same language, but within different intents and domains. Generative adversarial networks, paired with a transfer learning approach, can give us the opportunity to use other domains and intents which have robust data, to strengthen the performance of these low resource features. In this case we limit our experiments to exploring a case where we have a low resource domain. In mimicking this task, we pick one of the three domains we have as the low resource domain. We consider only the pseudo Goldens as available data for this domain. The remaining two domains are considered as robust domains and we consider all data (i.e., both pseudo Goldens and Rare utterances) as available data for the NLU model. We also use all training data from the two robust domains to pre-train word embedding using the fastText algorithm (Lample et al., 2017). We then run data generation for the low resource domain using our SeqGAN implementation while using the pre-trained word embeddings from the two robust domains to initialize word embeddings both in generator and discriminator. For the NLU model, we feed the pseudo Goldens together with the synthesized data for the low resource domain and feed all available training data for the two robust domains. We measure the performance of the low resource domain against a baseline which is run using only pseudo Goldens from the low resource domain together with all available data from the robust domains. As with task 1, performance is tested on the separate fixed test data set.

4 Data Generation

We explore the following different GAN frameworks and expansions to generate synthetic annotated utterances:

- **Original implementation with Monte Carlo rollout**: a selected set of experiments to benchmark the original implementation by Yu et al. (2017) on our open source data set.
- **Original implementation without rollouts**: a selected set of experiments to benchmark the original implementation by Yu et al. (2017), where the reward function is evaluated on a single MC rollout.

- **Generator with token-level reward**: training the generator using token-level reward as suggested by Hu et al. (2018).
- **Generator with token-level Monte Carlo rollout**: we expand the above implementation to include a token-level Monte Carlo rollout and test our task of bootstrapping new languages.
- **Generator with pre-trained embeddings**: we add fastText pre-trained embeddings to the generator and discriminator to selectively test our task of handling low-resource features).

The model architecture consists of three parts: generator pre-training, discriminator pre-training, and adversarial training. In the pre-training parts, we first train the generator, followed by the discriminator, for 80 epochs each. For each adversarial epoch, we update the generator once and then for 35 steps we generate negative examples using the current state of the generator, combined with the same number of positive examples from the training data, and re-train the discriminator. The total number of adversarial training epochs is set to 600. All hyper-parameters were chosen based on observations of when the loss functions and synthetic data quality evaluations either stabilize or clearly degrade.

5 Results and Discussion

5.1 Evaluation

We evaluate our models using domain accuracy, intent accuracy, slot F1, and frame accuracy. Domain and intent accuracy measure the accuracy of the domain and intent classification tasks, respectively. We use micro-averaging to calculate slot F1 to measure the performance of the NER task. Frame accuracy indicates the relative number of utterances for which the domain, intent, and all slots were correctly identified. In our case, we pay attention to individual metrics to understand which tasks are most affected by the synthesized data.

5.2 Language bootstrapping task

In this section we present the results of our experiment mimicking the task of bootstrapping a new language. See Section 3.3.1 for experimental setup. We compare three SeqGAN implementations each with a different reward policy. An open question when using generated data for NLU model training is whether the improvements observed are due to the data enrichment gained by the new variations introduced in synthetic data or due to upsampling. To test this, we repeat each experiment on three subsamples of the generated synthetic data. First, we add a synthetic data set that is equal in size to the pseudo Golden data set. We call this *TopX* sampling. This enables us to explore the changes in performance obtained by adding a synthetic data set that reflects the original distribution produced by the trained GAN framework, but with limited effects of upsampling. In the second case, we produce a synthetic data set that is significantly larger than the pseudo Golden data set (9600 utterances per domain) and we take only the unique utterances within that set. We call this *Uniques* sampling. This enables us to explore the changes in performance obtained by adding a synthetic dataset that contains an exhaustive set of the different combinations of utterances that the GAN can create, given the input data. Finally, we add the full set of 9600 generated utterances per domain, that should be similar in distribution to the pseudo Goldens set (*All* sampling). The results obtained for each SeqGAN implementation with TopX, Uniques, and All sampling strategies are summarized in Appendix B and Appendix C.

5.2.1 Generator with token-level reward

When using the Generator with token-level reward, we observe that out of three synthetic data sampling strategies (i.e., TopX, Uniques, and All), the biggest overall gain is shown by the All sampling strategy with an overall intent accuracy improvement of 4%, and an overall frame accuracy improvement of 3% (Table 1). Domain accuracy and overall slot F1 do not show much change. In this setup, alarm and reminder domains' intent accuracies show increases of 12-14% with overall intent accuracy increasing by 4%. However, intent accuracy in the weather domain degrades by 4%. The generator with token-level reward outperforms the corresponding generator with sentence-level reward in domain accuracy, slot F1 and frame accuracy, but trails in intent metrics.

When the models are run using Goldens upsampled to the same counts, we observe that overall domain, intent, and frame accuracies are slightly better. However, individual intent accuracy of alarm and reminder domains do not perform as well as with synthetic data. No degradation of the weather domain is observed with upsampled data.

Table 1: Performance relative to the baseline for models with token-level and sentence-level reward using All sampling strategy. Baseline is a model trained on Goldens only.

Reward	Domain	Intent accuracy			Overall	Slot F1			Overall	Frame
	accuracy	alarm	reminder	weather	int. acc.	alarm	reminder	weather	slot F1	accuracy
Token-level	0.29	11.84	14.62	-3.89	4.41	-1.04	0.8	1.25	0.39	3.26
Sent.-level	-0.26	14.45	17.4	0.27	7.81	-1.19	1.21	-2.77	-1.23	-4.43

Table 2: Performance relative to the baseline for models with Monte Carlo rollouts on token-level and sentence-level reward using Uniques sampling strategy. Baseline is a model trained on Goldens only.

Reward	Domain	Intent accuracy			Overall	Slot F1			Overall	Frame
	accuracy	alarm	reminder	weather	int. acc.	alarm	reminder	weather	slot F1	accuracy
Token-level	0.41	14.83	14.43	-0.12	7.21	-0.43	2.82	0.25	0.67	4.3
Sent.-level	-1.37	9.07	9.19	0.23	4.71	-0.09	1.47	-0.52	-0.12	0.19

5.2.2 Generator with token-level Monte Carlo rollout

Using a Generator with token-level Monte Carlo rollout brings significant improvement, especially outperforming other models in the Uniques sampling strategy. Overall intent accuracy shows an improvement of 7%, and overall frame accuracy improves by 4% (Table 2). Alarm and reminder domains' intent accuracy show improvements of 14-15%. The reminder domain's slot F1 improves by 3%. This setup performs significantly better than the generator with sentence-level reward in all overall metrics and also in alarm and reminder domain intent accuracies.

When compared to the models run using Goldens upsampled to the same counts, we observe that the Generator with token-level Monte Carlo rollout policy performs better on domain accuracy, overall intent accuracy, and overall slot F1 while being slightly under on frame accuracy. Specifically, it performs much better on both the alarm and reminder domains which show approximately 50% the performance boost.

5.3 Handling low resource features

In this section we present the results of our experiment mimicking the task of handling low resource features. See Section 3.3.2 for the experimental setup. We conduct each experiment three times and present the mean results for Uniques sampling strategy in Appendix D.

5.3.1 Generator with token-level reward and embeddings pre-trained on robust domains

Appendix D summarizes the results obtained using SeqGAN with a generator with token-level reward and embeddings pre-trained on robust domains to synthesize data for the low-resource domain. We observe a 12% increase in intent accuracy in the alarm domain when compared to the baseline. We also observe a 13% increase in intent accuracy in the reminder domain when synthetic data is added. For the weather domain, we do not observe a significant change in intent accuracy. For alarm and reminder domains we see an increase in overall intent accuracy when synthetic data is added. For these same domains, overall frame accuracy shows small improvements while domain accuracy and overall slot F1 does not show any significant changes.

5.3.2 Generator with token-level Monte Carlo rollout and embeddings pre-trained on robust domains

Appendix D shows the results obtained when using MC rollout policy in addition to using embeddings pre-trained on robust domains to synthesize data for the low-resource domain. For the reminder and alarm domains, we note that the performance boost in the IC task for the low-resource domain is larger by 2-3% than without the MC rollout. The weather domain shows a small but statistically significant improvement of 0.5% when compared to the baseline. These results suggest that the MC rollout policy provides additional guidance to the generator in all cases.

6 Synthetic data deep dive

6.1 Evaluating the quality of the data generated

To the best of our knowledge, there is no comprehensive metric that is commonly used for measuring the performance of a text generation model. To measure the performance of the SeqGAN, we

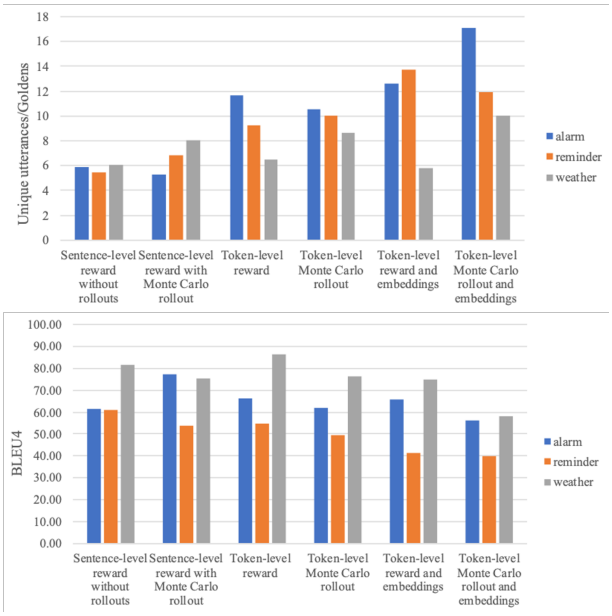


Figure 4: Data quality evaluation for synthetic data generated in different GAN models: number of unique utterances generated normalized by the number of Goldens used as an input for GAN model (left), and 4-gram BLEU score (right).

use the n-gram Bilingual Evaluation Understudy (BLEU) score (Papineni et al., 2002), calculated against a test set of Golden utterances. This metric measures the degree of similarity between the generated data and test set. Additionally, we calculate the diversity of generated data through the number of unique phrases we generated and number of unique words used from vocabulary. We also keep track on the mean utterance length to detect possible utterance collapse for token-level feedback, when the generator learns to generate shorter utterances. Detailed evaluations on bootstrapping experiments are provided in the Appendix E, and summarized in Figure 4. We see that compared to other domains, weather's BLEU scores are higher, and the diversity of the generated data is lower, suggesting that the model potentially reproduces almost the same data as the input, and does not bring in much novelty for NLU model training.

6.2 Annotation review

Although, in general, we observed that the NLU model benefits from synthetic data, we have noted some degradation in NER models, especially in the weather domain. Deep diving into NER errors, we found the following major sources of errors: annotation errors and context-dependent annotations. First, in seeding pseudo Goldens, we selected utterances based on their frequency, and then took all unique annotations to be a pseudo Goldens set. That process artificially increased weight of annotation errors in cases were for frequent phrases there were a few misannotated utterances. For example, for utterances "*remind me to ... tomorrow*" with and without label "*datetime*" frequently appeared in our goldens, and the NLU model fails to recognize "*tomorrow*" as an entity. In contrast, for the utterances "*remind me tomorrow to ...*" the model produces the correct "*datetime*" label. Additionally, small context-dependent words that have different annotation in the same domain, but appear to have a dominant annotation in pseudo Goldens and further in synthetic data, happen to be another cause of failures. One example of such a word is "*for*" in the weather domain, where in "*weather for London next week*" it has no label, while in "*forecast for next week please*" it is labeled as datetime.

7 Conclusions

In this paper, we evaluate the use of the SeqGAN model for synthetic annotated data generation to boost NLU model performance. We have shown that adding synthetic data to bolster our Goldens can significantly improve DNN model performance in intent classification and named entity recognition tasks. We propose a token-level reward with Monte Carlo search rollout to guide the generator model, that showed better performance when compared with a regular token-level reward implementation, sentence-level reward implementations both with and without Monte Carlo tree search, and with a pure upsampling strategy. We also show that using SeqGAN together with embeddings pre-trained on high-resource domains to generate synthetic data can significantly improve the performance of low-resource domains. Embeddings pre-trained on different tasks can carry over the information they have learned and that can be especially useful in low-resource model building scenarios.

References

Tong Che, Yanran Li, Ruixiang Zhang, R. Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. 2017. Maximum-likelihood augmented discrete generative adversarial networks. *ArXiv*, abs/1702.07983.

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

deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

William Fedus, Ian J. Goodfellow, and Andrew M. Dai. 2018. Maskgan: Better text generation via filling in the _. *ArXiv*, abs/1801.07736.

Ge Gao, Eunsol Choi, Yejin Choi, and Luke Zettlemoyer. 2018. Neural metaphor detection in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 607–613, Brussels, Belgium. Association for Computational Linguistics.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680. Curran Associates, Inc.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long text generation via adversarial training with leaked information. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, pages 5141–5148, New Orleans, Louisiana, USA. AAAI Press.

Rahul Gupta. 2019. Data augmentation for low resource sentiment analysis using generative adversarial networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7380–7384. IEEE.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1587–1596, International Convention Centre, Sydney, Australia. PMLR.

Zhiting Hu, Zichao Yang, Tiancheng Zhao, Haoran Shi, Junxian He, Di Wang, Xuezhe Ma, Zhengzhong Liu, Xiaodan Liang, Lianhui Qin, Devendra Singh Chaplot, Bowen Tan, Xingjiang Yu, and Eric Xing. 2018. Texar: A modularized, versatile, and extensible toolbox for text generation. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 13–22, Melbourne, Australia. Association for Computational Linguistics.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *ArXiv*, abs/1312.6114.

Varun Kumar, Hadrien Glaude, Cyprien de Lichy, and Wlliam Campbell. 2019. A closer look at feature space data augmentation for few-shot intent classification. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 1–10, Hong Kong, China. Association for Computational Linguistics.

Matt J. Kusner and José Miguel Hernández-Lobato. 2016. Gans for sequences of discrete elements with the gumbel-softmax distribution. *ArXiv*, abs/1611.04051.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *ArXiv*, abs/1711.00043.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *ArXiv*, abs/1701.06547.

Kevin Lin, Dianqi Li, Xiaodong He, Ming-Ting Sun, and Zhengyou Zhang. 2017. Adversarial ranking for language generation. In *Advances in Neural Information Processing Systems 30*, pages 3155–3165. Curran Associates, Inc.

Jie Lu, Vahid Behbood, Peng Hao, Hua Zuo, Shan Xue, and Guangquan Zhang. 2015. Transfer learning using computational intelligence: A survey. *Knowledge-Based Systems*, 80:14–23.

Nikolaos Malandrakis, Minmin Shen, Anuj Goyal, Shuyang Gao, Abhishek Sethi, and Angeliki Metallinou. 2019. Controlled text generation for data augmentation in intelligent artificial agents. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 90–98, Hong Kong. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA. Association for Computational Linguistics.

Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, Eric King, Kate Bland, Amanda Wartick, Yi Pan, Han Song, Sk Jayadevan, Gene Hwang, and Art Pettigrue. 2018. Conversational AI: The science behind the alexa prize. *ArXiv*, abs/1801.03604.

Sebastian Schuster, Sonal Gupta, Rushin Shah, and Mike Lewis. 2019. Cross-lingual transfer learning for multilingual task oriented dialog. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3795–3805, Minneapolis, Minnesota. Association for Computational Linguistics.

Burr Settles. 2009. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Richard S Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*, pages 1057–1063. MIT Press.

Ziang Xie, Sida I. Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y. Ng. 2017. Data noising as smoothing in neural network language models. *ArXiv*, abs/1703.02573.

Jingjing Xu, Xuancheng Ren, Junyang Lin, and Xu Sun. 2018. Diversity-promoting gan: A cross-entropy based generative adversarial network for diversified text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3940–3949, Brussels, Belgium. Association for Computational Linguistics.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*, pages 2852–2858. AAAI Press.

Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. 2017. Adversarial feature matching for text generation. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 4006–4015, International Convention Centre, Sydney, Australia. PMLR.

BertAA: BERT fine-tuning for Authorship Attribution

Maël Fabien[1,2], **Esaú Villatoro-Tello**[1,3], **Petr Motlicek**[1], and **Shantipriya Parida**[1]

[1]Idiap Research Institute, Martigny, Switzerland.
{firstname.lastname}@idiap.ch
[2]Ecole Polytechnique Fédérale de Lausanne, Switzerland.
mael.fabien@epfl.ch
[3]Universidad Autónoma Metropolitana, Unidad Cuajimalpa, Mexico City, Mexico.
evillatoro@correo.cua.uam.mx

Abstract

Identifying the author of a given text can be useful in historical literature, plagiarism detection, or police investigations. Authorship Attribution (AA) has been well studied and mostly relies on a large feature engineering work. More recently, deep learning-based approaches have been explored for Authorship Attribution (AA). In this paper, we introduce BertAA, a fine-tuning of a pre-trained BERT language model with an additional dense layer and a softmax activation to perform authorship classification. This approach reaches competitive performances on Enron Email, Blog Authorship, and IMDb (and IMDb62) datasets, up to 5.3% (relative) above current state-of-the-art approaches. We performed an exhaustive analysis allowing to identify the strengths and weaknesses of the proposed method. In addition, we evaluate the impact of including additional features (e.g. stylometric and hybrid features) in an ensemble approach, improving the macro-averaged F1-Score by 2.7% (relative) on average.

1 Introduction

Authorship Analysis is the field of Natural Language Processing that studies the characteristics of a text and extracts information on its author. It is made of 3 sub-tasks, which include author profiling, i.e. detecting sociolinguistic attributes such as gender or age, authorship verification which identifies the degree of similarity of texts, and authorship attribution (El et al.). Authorship Attribution (AA) is the process of attributing a text to the correct author among of closed set of potential writers. AA is widely used in plagiarism detection or attribution of historical literature (Li). This classification task is also well known in forensic investigations (Yang and Chow, 2014).

AA has been studied on short texts (Aborisade and Anwar, 2018), such as Tweets as well as longer texts, such as judgments of a few thousand words on average (Sari et al., 2018). The main challenge in AA is the extraction of relevant features characterising the author's identity. Majority of approaches proposed in the past relied on a large amount of feature engineering, in order to reflect both the content and the style of the author (Madigan et al., 2005; Aborisade and Anwar, 2018; Seroussi et al., 2014; Bozkurt et al., 2007).

In this paper, we propose a method, BertAA, that relies on the fine-tuning of a pre-trained BERT language model, to which we add a dense layer and a softmax activation for authorship classification, trained for a few epochs. This is one of the very first attempts to analyze the performances of pre-trained language model fine-tuning for in-domain AA, especially for a large number of authors (up to 100). As most Deep-Learning methods for AA, BertAA does not require text preprocessing nor feature engineering. Our method offers state-of-the-art (SOTA) performances on well-known corpora, with a relative accuracy improvement of up to 5.3%. We also illustrate the strengths and weaknesses of such a system. We also show that building an ensemble architecture, which also incorporates stylometric and hybrid features tends to improve the macro-averaged F1-score. Finally, we set a benchmark for the full IMDb corpus (Seroussi et al., 2014) for 5, 10, 25, 50, 75, and 100 authors, which, to the best of our knowledge, has never been studied in its full format for AA.

The next section discusses the relevant approaches developed in the literature. Section 3 presents the corpora used as well as a brief exploration of each of the sources. Section 4 details the architectures of BertAA, while Section 5 describes the results we obtained, and Section 6 discusses the strengths and weaknesses of our method, as well as future work directions. Finally, Section 7 depicts our conclusions.

Proceedings of the 17th International Conference on Natural Language Processing, pages 127–137
Patna, India, December 18 - 21, 2020.

2 Related work

Traditionally, AA largely relies on the process of extracting features related to content or style of an author (Stamatatos, 2009). More recently, some approaches propose to use deep learning methods for AA tasks, whether relying on a previous feature extraction step or not. The following sections briefly describe these various methods.

2.1 Traditional methods

Term Frequency - Inverse Document Frequency (TF-IDF) is used in AA at the word or the word or character N-gram level. It captures the words, the stems, or the combinations of words or letters that an author uses. Some recent works combine the votes of several classifiers on several levels of N-grams (Muttenthaler et al.). Such methods are referred to in the literature as being content-related classifiers (Sari et al., 2018).

In addition, stylometric features reflect the style of the author (Sari et al., 2018). The main hypothesis behind this feature extraction is that each author has its own writing style (e.g use of punctuation, average word length, sentence length, number of upper cases...). Features reflecting the style are used as an input for a LR usually, as seen in (Madigan et al., 2005; Aborisade and Anwar, 2018; Madigan et al.). An optional step of text pre-processing is often added (Allison and Guthrie), and more specifically through stop-words removal and stemming. Sari et al. (2018) reached an accuracy of 95.9% on the IMDb62 dataset (Seroussi et al., 2014), 1.1% (absolute) above a character N-gram classifier, by including stylometric features in a classifier. Soler-Company and Wanner (2017) also showed that including syntactic and discourse features can help achieve SOTA performances in author and gender identification.

To combine the numerous sources of input features, AA is also performed using ensemble learners, made for example of several SVM classifiers (Bacciu et al., 2020). Each classifier is trained on certain features related to distinct concepts, such as style, content, author profiling, etc.

2.2 Deep Learning based methods

While overcoming the burden of feature engineering, deep learning-based methods have reached SOTA results, whether through the use of Long Short-Term Memory (LSTM) (Qian et al.) at both the sentence and article-level or using multi-headed Recurrent Neural Network (RNN) (Bagnall, 2016) for on short multi-lingual texts. Convolutional Neural Networks (CNN) have also been widely explored for AA and can extract information from raw signals in speech processing or computer vision.

Ruder et al. (2016) explored CNNs at the word and character level for AA and found that CNNs at the character level tend to outperform other simple approaches based on SVMs for example, while CNNs at the N-gram level have been shown to perform competitively (Shrestha et al., 2017). Zhang et al. (2018) proposed a Syntax-augmented CNN model which outperforms other approaches on the Blog authorship and the IMDb62 datasets.

Siamese networks are well known in computer vision, e.g. for facial recognition tasks (Wu et al., 2017). Saedi and Dras (2019), used Convolutional Siamese Networks to perform AA. They compared their approach with a BERT fine-tuning over 3 epochs and showed that Siamese Networks are more robust over large-scale AA tasks ($N > 50$). This type of approach has the advantage of being able to evaluate the similarity between texts, as shown in (Qian et al.).

In 2020, Barlas and Stamatatos (2020) leveraged pre-trained language models (BERT, ELMo, ULMFiT, GPT-2) for the specific case of cross-topic and cross-domain AA on the CMCC dataset (Goldstein-Stewart et al., 2009), on a subset of 21 authors. The authors used a multi-headed classifier with a demultiplexer. In an N-authors classification task (N typically < 100), N classifiers would be trained, each of them seeing predominantly data from one author. In prediction, the text to classify is passed through all classifiers, and after normalization, the scores are compared. This work shows that BERT seems to work best on large vocabularies, and outperforms multi-headed RNNs.

Contrary to previous work, in this paper we perform an exhaustive analysis on the performance of pre-trained language models, we identify the advantages and limitations on three well-known benchmark datasets. In addition, we evaluate the impact of incorporating stylometric and hybrid features through ensemble techniques.

3 Authorship Attribution Corpora

Several corpora have been studied for the task of AA. In this section, we briefly describe each corpus we used as well as its key features. The AA task

we performed focuses on identifying the author of a text among a list of the top N authors for whom we collected the largest number of texts.

3.1 Enron Email corpus

Enron Email corpus has been widely studied over the previous decade since the bankruptcy of Enron. 517'401 emails from around 160 employees were made public, and data preparation for email classification was then done by Klimt and Yang (2004). The emails mainly contain conversations of managers at Enron, and given the fraudulent nature of the emails, it is commonly used as a study case for criminal network investigations (Aven, 2015).

Emails were collected from the "Sent" folder of each of the 160 employees. Since around 13% of the emails contained the name of the sender, as a signature or side information in a forwarded message, we dropped these observations. We also removed all messages of less than 10 tokens to apply the same processing as Ruder et al. (2016). Our end corpus contained 130'000 emails. Emails are on average 150 tokens long, and the median length is 61 tokens.

Enron Email corpus has already been studied for several Authorship Analysis tasks, including Authorship Verification (Halvani et al., 2020; Brocardo et al., 2013), as well as for AA tasks (Neumann and Schnurrenberger; Li; Allison and Guthrie). Gender identification and sentiment analysis were also studied by Clough et al. (2011).

3.2 IMDb Authorship Attribution Corpus

The IMDb Authorship Attribution corpus was introduced by Seroussi et al. (2014). 271'000 movie reviews were produced by 22'116 distinct authors, with an average of 12.3 texts per author. Texts are on average 121 tokens long. No preprocessing or filtering was applied to the corpus.

Most of the works that we have found referred to the IMDb62 dataset, a truncated version of the IMDb Authorship Attribution Corpus with 62 authors and 1'000 texts per author. We chose to benchmark our solution on the IMDb62 against other approaches, but also to evaluate the performance of our model on the full version of the corpus since it contains a class imbalance (closer to a real-life scenario) and has more data, with an average of 3'900 texts per author for the top 5 authors. The full version of the corpus has, to the best of our knowledge, never been studied for AA for a various number of authors. Hence, our approach sets a benchmark.

3.3 Blog Authorship Attribution Corpus

The Blog Authorship Attribution corpus is a corpus of blog articles from 2004 and before, collected from blogger.com. It was introduced by Schler et al. as part of a study on the effects of age and gender on blogging. More than 680'000 posts are available, from more than 19'000 authors. An average of 35 posts was collected per author. No preprocessing or filtering was applied to the corpus. Although it might seem surprising, it is worth mentioning that this dataset is the one containing the shortest texts on average (79 tokens for the top 5 authors, vs 190 for Enron). Many of the blog posts collected were replies to existing blog posts or short articles.

For our experiments, we considered the top 5, 10, 25, 50, 75, and 100 authors with the largest number of texts. Table 1 presents the summary statistics of the length and number of documents per author, in the various configurations considered, for each dataset. As a summary, Enron has rather long texts, a large number of texts per author with a large associated standard deviation. IMDb reviews are shorter and the number of texts per author is lower than for Enron. Finally, for the Blog dataset, the texts are short, and the number of texts per author is smaller than for Enron, with fewer variability than for IMDb.

Dataset	N	Avg. Num. Tokens	Avg. Nb. Texts
Enron	5	190 (± 375)	11205 (± 2324)
	10	201 (± 419)	8745 (± 3052)
	25	185 (± 375)	5626 (± 3230)
	50	183 (± 361)	3685 (± 3014)
	75	194 (± 386)	2774 (± 2779)
	100	208 (± 717)	2259 (± 2567)
IMDb	5	106 (± 184)	3900 (± 2197)
	10	127 (± 185)	2817 (± 1895)
	25	110 (± 167)	1873 (± 1434)
	50	104 (± 152)	1324 (± 1155)
	75	102 (± 158)	1080 (± 1005)
	100	102 (± 157)	932 (± 907)
IMDb62	62	341 (± 223)	1000 (± 0)
Blog	5	79 (± 191)	2659 (± 780)
	10	91 (± 184)	2350 (± 639)
	25	99 (± 174)	1832 (± 599)
	50	98 (± 167)	1466 (± 562)
	75	120 (± 209)	1270 (± 538)
	100	126 (± 228)	1122 (± 533)

Table 1: Descriptive statistics for the 4 datasets. N: number of authors, Avg. Num. Tokens: average number of tokens per text, Avg. Nb. Texts: average number of texts. Standard deviation in parenthesis.

4 BertAA : BERT-based Authorship Attribution

Content-related features in AA take into account the topics and the semantics of the text. Recent works on language representation models have however shown that transformers such as BERT (Devlin et al., 2019) reach SOTA performances for various tasks, hence improving GLUE score as well as several other metrics. It has been extensively used for text classification tasks (Sun et al., 2020), and BERT is known to be well-performing at extracting semantic and syntactic information.

To the best of our knowledge, no systematic review of the performance of fine-tuned pre-trained language models for AA has been reported yet, and such classifier has never been combined with a stylometric and hybrid features in an ensemble model. Hereby, we introduce BertAA, a fine-tuning of BERT with a dense layer and a softmax activation, trained for a few epochs for AA. The output dimension of the dense layer corresponds to the number of authors in the corpus.

BERT is made of 12 Transformer blocks and 12 self-attention heads. The input size, i.e. the maximum length of tokens is 512, and the hidden layer representation dimension is 768 (Vaswani et al., 2017). As described by Sun et al. (2020), to use BERT as a classifier, a simple dense layer with softmax activation is added on top of the final hidden state h of the first token [CLS], through a weight matrix W, and we predict the probability of label c the following way:

$$p(c \mid \mathbf{h}) = \text{softmax}(W\mathbf{h}). \quad (1)$$

Then, all weights, including BERT's ones and W, are adapted, in order to maximize the log-probability of the correct label. The training is done using a Cross-Entropy loss function. We used a pre-trained BERT available from the Transformer library (Wolf et al., 2020), trained on large corpora. The fine-tuning of BERT for the AA task was done on a Tesla P100-PCIE-16GB.

Additionally, we incorporate stylometric and hybrid features to BertAA, in 2 models called BertAA + Style and BertAA + Style + Hybrid through a LR. Thus, our system is able to account for content, stylometric, and hybrid features. The architecture of BertAA + Style + Hybrid is presented in Figure 1.

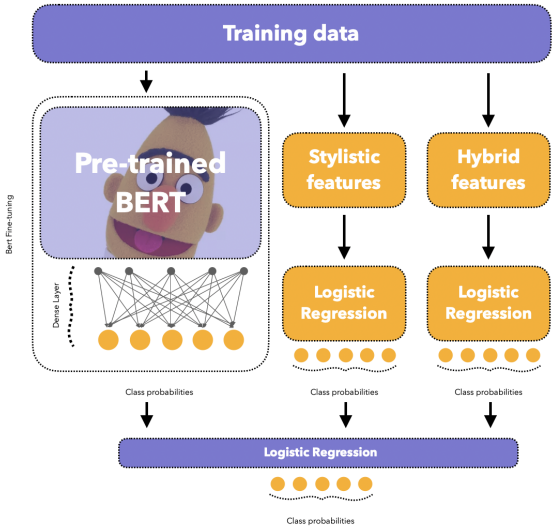


Figure 1: BertAA + Style + Hybrid architecture.

The stylometric classifier first extracts the lexical stylometric features as proposed by Sari et al. (2018). The features extracted are the length of text, the number of words, the average length of words, the number of short words, the proportion of digits and capital letters, individual letters and digits frequencies, hapax-legomena, a measure of text richness, and the frequency of 12 punctuation marks. A LR is trained on these features. The hybrid features we extract are the frequencies of the 100 most frequent character-level bi-grams and tri-grams. Classification is then done using a LR. Finally, the output probabilities of Bert classifier, the stylometric, and the hybrid ones are concatenated and classified using an additional LR.

5 Results

Parameters we chose for our architectures are presented in Table 2.

Model	Parameter	Value
Hybrid feat.	Char. N-grams	(2,3)
LR	Penalty	l2
	Tolerance	0.0001
	C	1.0
	Max Iterations	100
	Intercept	True
BERT	Config	bert-base-cased
	Epochs	1 to 5
	Input token length	512

Table 2: Parameters of the experiments.

We ran the experiments on 5, 10, 25, 50, 75, and 100 authors for the full IMDb, the Blog, and Enron datasets presented above. Our model was trained

on 5 epochs for each experiment unless specified otherwise. The results are presented in Table 3. We picked the top N authors with the largest amount of texts, for each of the datasets, and kept 20% of test data using a stratified approach, meaning that the proportions of each class are kept equal in the training and testing set. We report the results of BertAA, BertAA + Style and BertAA + Style + Hybrid. We compare our approach with a word-level TF-IDF - LR model with stemming and stop-words removal. We also add as a benchmark the performance of a LR trained only on stylometric features, and an additional LR trained on the character-level N-gram hybrid features.

BertAA outperforms the TF-IDF and LR benchmark on all experiments, with an average relative accuracy gain of 14.3%. It reaches a competitive performance on 5 authors on the Enron dataset, since only 2 samples were not classified correctly out of 4104, hence leading to an accuracy of 99.95%.

Comparing results on Enron to other approaches in literature is not trivial since it largely depends on the data preparation that was done. We decided to remove short emails, and remove utterances containing the name of the sender (as a signature for example), but not all papers involving Enron data for AA precisely describe their data preparation. Furthermore, we found no results in the literature on IMDb full-corpus for the top N authors. Hence, our results set a benchmark on the full IMDb, on average 8.2% above a word-level TF-IDF. Next, we compare our results with current SOTA on the IMDb62 and the Blog Authorship datasets.

5.1 How does the performance compare to SOTA?

In Table 4, we report the accuracy of our best systems (no additional features, 5 epochs) on the Blog Authorship corpus against the performances of several CNN-based architectures, including the character-level CNN presented in (Ruder et al., 2016) and current SOTA Syntax-enriched CNN(Zhang et al., 2018). We report results over 10 and 50 authors. For 10 authors, the accuracy of our best BertAA system (no additional features, 5 training epochs) reaches 65.4% which is, to the best of our knowledge, the current SOTA on the Blog Authorship Corpus, and represents a relative improvement of 2% over the Syntax CNN. When the number of authors increases, our system displays an accuracy of 59.7%, which represents a relative improvement of 5.3% accuracy compared to the previous SOTA. The main characteristics of the Blog Corpus are that texts are rather short on average (respectively 91 and 98 tokens on average for 10 and 50 authors), while the number of texts per authors remains quite high on average, with a rather small standard deviation, suggesting that BertAA is well suited for datasets with short sentences, and a large but balanced number of texts per author.

Approach	10	50
Impostors (Koppel and Winter, 2014)	35.4	22.6
SCAP (Frantzeskou et al., 2006)	48.6	41.6
LDAH-S (El et al.)	52.5	18.3
CNN (Ruder et al., 2016)	61.2	49.4
Continuous N-gram (Sari et al., 2017)	61.3	52.8
N-gram CNN (Zhang et al., 2018)	63.7	53.1
Syntax CNN (Zhang et al., 2018)	64.1	56.7
BertAA	**65.4**	**59.7**

Table 4: Accuracy on Blog Authorship

5.2 Are external features useful?

In order to assess the impact of external features, we compute the accuracy per author on the Blog dataset for 10 authors. We compare the per-author accuracy of a word-level TF-IDF + LR classifier and BertAA, to identify whether TF-IDF outperforms our system on some classes in Figure 2.

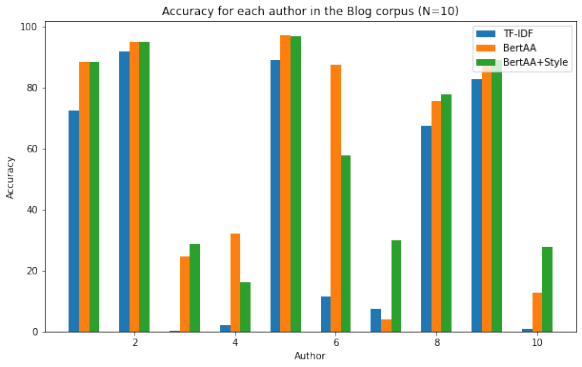


Figure 2: Accuracy per author for TF-IDF and BertAA (+Style) on the Blog Dataset (N=10)

On most authors, BertAA slightly outperforms TF-IDF, although both methods reach good accuracies. However, BertAA brings additional value, especially where TF-IDF performs poorly, e.g. on Author 3 in the figure. In some specific cases, such as for author "7" on the figure, TF-IDF achieves a better performance than BertAA. In such a case, adding the stylometric features improves the per-

Dataset	N-Authors	Baslines			Proposed Method		
		Stylo.	Char N-gram	TF-IDF	BertAA	+ Style	+ Style + Hybrid
Enron	5	75.0	84.4	98.0	**99.95**	**99.95**	**99.95**
	10	54.9	70.5	96.4	**99.1**	**99.1**	**99.1**
	25	35.6	53.2	92.7	**98.7**	**98.7**	**98.7**
	50	20.4	44.8	90.8	98.1	**98.2**	**98.2**
	75	17.3	40.6	90.1	**97.6**	97.5	97.5
	100	15.8	36.9	88.3	97.0	97.0	**97.1**
IMDb	5	65.8	92.1	98.1	**99.6**	**99.6**	**99.6**
	10	44.6	79.2	93.9	98.1	**98.2**	**98.2**
	25	25.5	55.8	84.1	**93.2**	92.9	92.9
	50	17.4	44.2	82.1	**90.7**	90.6	90.6
	75	14.7	37.6	79.2	**88.3**	87.8	87.8
	100	11.8	33.6	76.6	**86.1**	85.3	85.4
Blog	5	34.7	40.0	45.7	**61.3**	59.7	59.8
	10	18.9	31.9	45.0	**65.4**	62.4	62.4
	25	9.9	23.4	42.0	**65.3**	64.4	64.4
	50	6.2	15.7	41.4	**59.7**	58.7	58.7
	75	5.0	15.7	42.2	**60.9**	59.0	59.2
	100	4.2	13.8	40.5	**58.8**	57.3	57.6

Table 3: Accuracy on the number of authors for all approaches on the 3 datasets.

formance of our model on this author. But what is the overall impact of additional features on the model performance?

Adding stylometric and hybrid features in the first experiment on the Blog corpus, with 10 authors, the accuracy decreases from 65.4 to 62.4%. However, the macro-averaged F1-score we report using these features is higher, at 61.4% instead of 56.7% when no features are added.

This behavior of BertAA is illustrated in the confusion matrices in Figure 3, in which we report the accuracy per class (i.e. per author). Surprisingly, BertAA is stuck at 0% accuracy on certain authors, as it tends to allocate all the texts to a sub-set of authors, which can lead to a good accuracy but a lower macro-averaged F1-score. On the other hand, adding other features (stylometric and hybrid) improves the macro-averaged F1-score, but reduces the accuracy in that specific case.

According to our experiments, as illustrated in Figure 4, the F1-score on the Blog Authorship corpus for 5, 10, 25, 50, 75, and 100 authors improves by 2.70% (relative) when stylometric features are added to BertAA, and by 2.73% (relative) when including hybrid features.

In the blog corpus, more than 2'300 texts are collected per author, for the top 10 authors, which offers a sufficient quantity of training data, and a limited number of authors. But this does not guarantee that our model behaves well under a smaller set of training data and a wider classification task, such as on the IMDb62 dataset.

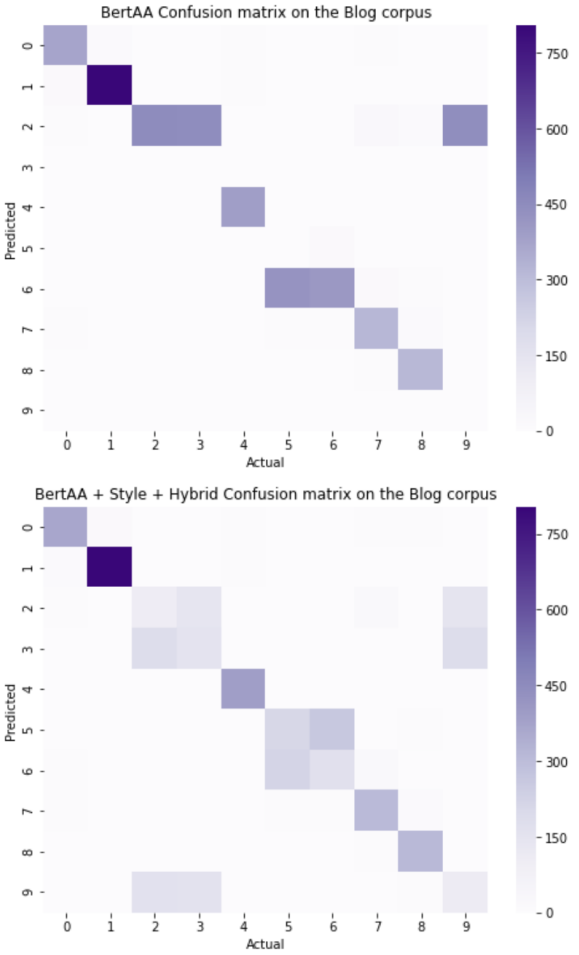


Figure 3: Confusion matrix of BertAA and BertAA + Style + Hybrid on the Blog corpus

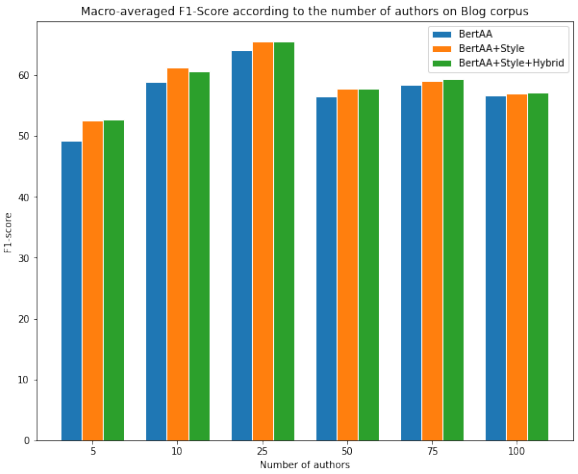


Figure 4: Macro-averaged F1-score when including stylometric and hybrid features according to number of authors.

5.3 More authors, less data

We ran additional experiments on the IMDb62 dataset with a larger number of authors (62), and fewer training samples per author (1'000). To replicate the setup of most methods presented in Table 5, 20% of the data were used as a test sample. The split is made randomly, since no standardized training and testing corpus exists for all these datasets. In Table 5, we report the performance of the various BertAA architectures and compare our approaches to various methods including current SOTA.

Approach	Accuracy
LDA+Hellinger (El et al.)	82
Word Level TF-IDF	91.4
CNN-Char (Ruder et al., 2016)	91.7
Comp.Att.+Sep.Rec. (Song et al., 2019)	91.8
Token-SVM (Seroussi et al., 2014)	92.52
SCAP (Frantzeskou et al., 2006)	94.8
Cont. N-gram Char (Sari et al., 2017)	94.8
(C+W+POS)/LM (Kamps et al., 2017)	95.9
N-gram + Style (Sari et al., 2018)	95.9
Syntax CNN(Zhang et al., 2018)	**96.2**
BertAA + Style + Hybrid - 1 epoch	88.7
BertAA + Style - 3 epochs	91.1
BertAA + Style + Hybrid - 5 epochs	92.3
BertAA + Style + Hybrid - 10 epochs	93.0

Table 5: Accuracy of various approaches on IMDb62

The Syntax-enriched CNN presented in (Zhang et al., 2018) reached an accuracy of 96.2%. Most other approaches lie between 91 and 94%. Considering that IMDb62 offers 1'000 training samples per author, the training of BertAA over a single epoch did not perform well. We then increased the number of training epochs and reached 92.3% at 5 epochs, and up to 93.0% at 10 epochs. This highlights the limitations of our model in situations with less training data and more authors.

Figure 5 plots the relative accuracy of BertAA over the number of authors for all three datasets. The starting point at 100 represents the accuracy reached by the model at 5 authors. A decreasing trend would therefore illustrate that the model accuracy is negatively impacted by a larger number of authors. On Enron and the Blog, the decrease in accuracy is limited, since 95 to 97% of the performance on 5 authors is maintained at 100 authors. The largest decrease occurs for IMDb dataset, at around 87% of the accuracy at 5 authors for 100 authors. This can likely be explained by the fact that IMDb comments are published publicly on the IMDb website, and that many authors might read comments of a movie before publishing theirs. Words, topics, punctuation, or phrases might therefore be re-used by some authors when publishing their comments.

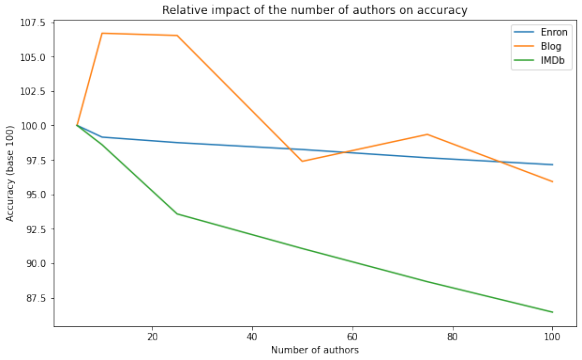


Figure 5: Relative impact of the number of authors on accuracy, base being the case with 5 authors.

Since the accuracy increases over the number of epochs on IMDb 62, in the next section, we further explore the impact of the number of training epochs on the model performance.

5.4 How much fine-tuning is too much?

In literature, the recommended number of epochs for BERT has been set between 2 and 4 (Sun et al., 2020), 3 being a common choice. In order to explore the effect of the number of training epochs on the model's accuracy, we report in Figure 6 the accuracy for the IMDb62 dataset using several models (BertAA, BertAA + Style, BertAA + Style + Hybrid), and compare it to the baseline TF-IDF. BertAA + Style appears to be the best performing model and starts to offer better performances

than TF-IDF after 4 epochs. However, no peak performance is reached, and the accuracy is still improving after 10 epochs, although to a lesser extent. The impact of the training epochs on a dataset with 62 authors is higher since BertAA is not performing as well, but we can suppose that the impact of the number of epochs is reduced on a smaller set of authors. We have chosen to train our models on 5 epochs for most of our experiments since it offers a good tradeoff between the training time and accuracy.

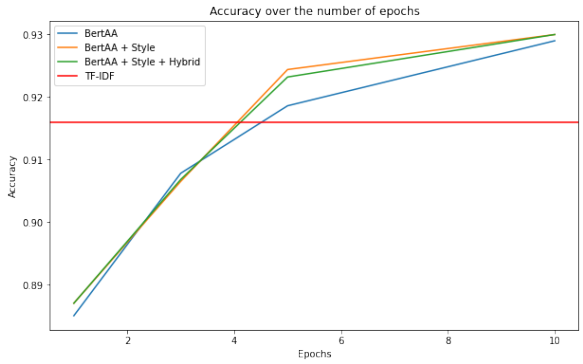


Figure 6: Accuracy over the number of epochs

6 Discussion

Our approach can be summarized as an extrapolation of BERT's general outstanding scores, for AA. We show that reaching SOTA results can be achieved using only a single dense layer and a softmax activation on top of a pre-trained BERT with a few training epochs. We highlighted that BertAA performs well on rather short texts, few imbalances in the number of texts per author, and a large number of texts per author.

Previous works (Sari et al., 2018) have shown that using stylometric and hybrid features improves the accuracy of AA tasks. We also show that adding such features when leveraging pre-trained language models can improve the macro-averaged F1-Score by 2.7% (relative) on average, although impacting the accuracy.

The use of BertAA should be limited to cases where BERT is itself a good candidate, i.e. when there are sufficient training data per author. This condition might be hard to reach in real applications for police investigations. Short texts and few imbalances have also been identified as requirements for better model performances. Our model is also currently unable to perform text similarity evaluation in the context of Authorship Verification.

There are many possible extensions to this work. According to our experiments on Enron, Blog Authorship, and IMDb corpora, AA can successfully leverage transformers-based language representation models. So far, we have not performed further pre-training of BERT on the target domain, which could also help BertAA. We have not tried yet to use another pre-trained language model. Future works should also explore other model architectures like RoBERTa(Liu et al., 2019), or try to extract additional stylometric, hybrid, profiling, or content-related features. Including the computation of similarity metrics on embeddings learned through the BERT fine-tuning would also be a way to compare the similarity between texts for Authorship Verification tasks. We will also explore BertAA in AA tasks on ASR transcripts, where punctuation and capital letters are not present for example.

7 Conclusion

With the rise of Deep Learning and Transformers in Natural Language Processing, feature engineering and text preprocessing are less needed.

In this work, we presented an approach based on fine-tuning of a pre-trained BERT for author classification. This is one of the very first attempts to analyze the performances of pre-trained language model fine-tuning for in-domain AA. We showed that our approach, which leverages BERT, reaches competitive performances on three well-known benchmark datasets, even on a large number of authors. The model best performs when sufficient training data per author are available, there is no large class imbalance, and texts remain rather short.

We also show that in a large scale AA task, adding stylometric and hybrid features to BertAA in an ensemble model can improve the macro-averaged F1-score by 2.7% (relative) on average. Finally, we set a new benchmark on the full IMDb Authorship Attribution Corpus for 5, 10, 25, 50, 75, and 100 authors. Future works will explore adding features to BertAA, further pre-training BERT on target-domain, exploring other pre-trained language models, and extending our approach to Authorship Verification. [1]

[1] Code and datasets are available here

Acknowledgment

This work was supported by the European Union's Horizon 2020 research and innovation program under grant agreement No. 833635 (project ROXANNE: Real-time network, text, and speaker analytics for combating organized crime, 2019-2022). The second author, Esaú Villatoro-Tello, was supported partially by Idiap, SNI-CONACyT, CONACyT project grant CB-2015-01-258588, and UAM-C Mexico during the elaboration of this work.

References

Opeyemi Aborisade and Mohd Anwar. 2018. Classification for Authorship of Tweets by Comparing Logistic Regression and Naive Bayes Classifiers. In *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, pages 269–276.

Ben Allison and Louise Guthrie. Authorship Attribution of E-Mail: Comparing Classifiers Over a New Corpus for Evaluation. page 5.

Brandy L. Aven. 2015. The Paradox of Corrupt Networks: An Analysis of Organizational Crime at Enron. *Organization Science*, 26(4):980–996. Publisher: INFORMS.

Andrea Bacciu, Massimo La Morgia, Alessandro Mei, Eugenio Nerio Nemmi, and Julinda Stefa. 2020. Cross-Domain Authorship Attribution Combining Instance-Based and Profile-Based Features. page 14.

Douglas Bagnall. 2016. Author Identification using Multi-headed Recurrent Neural Networks. *arXiv:1506.04891 [cs]*. ArXiv: 1506.04891.

Georgios Barlas and Efstathios Stamatatos. 2020. Cross-Domain Authorship Attribution Using Pre-trained Language Models. In Ilias Maglogiannis, Lazaros Iliadis, and Elias Pimenidis, editors, *Artificial Intelligence Applications and Innovations*, volume 583, pages 255–266. Springer International Publishing, Cham. Series Title: IFIP Advances in Information and Communication Technology.

İlker Nadi Bozkurt, Özgür Bağlıoğlu, and Erkan Uyar. 2007. Authorship attribution: performance of various features and classification methods. In *22nd International Symposium on Computer and Information Sciences, ISCIS 2007 - Proceedings*, pages 158–162. IEEE. Accepted: 2016-02-08T11:41:59Z.

Marcelo Luiz Brocardo, Issa Traore, Sherif Saad, and Isaac Woungang. 2013. Authorship verification for short messages using stylometry. In *2013 International Conference on Computer, Information and Telecommunication Systems (CITS)*, pages 1–6, Athens, Greece. IEEE.

Paul Clough, Colum Foley, Cathal Gurrin, Gareth Jones, Wessel Kraaij, Hyowon Lee, and Vanessa Murdock, editors. 2011. *Advances in Information Retrieval: 33rd European Conference on IR Resarch, ECIR 2011, Dublin, Ireland, April 18-21, 2011, Proceedings*. Information Systems and Applications, incl. Internet/Web, and HCI. Springer-Verlag, Berlin Heidelberg.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*. ArXiv: 1810.04805.

Sara El, Manar El Bouanani, Ensias Mohammed, Mohammed Ben, Abdallah Regragui, and Madinat Al. *General Terms Authorship analysis*.

Georgia Frantzeskou, Efstathios Stamatatos, Stefanos Gritzalis, and Sokratis Katsikas. 2006. Source Code Author Identification Based on N-gram Author Profiles. In *Artificial Intelligence Applications and Innovations*, IFIP International Federation for Information Processing, pages 508–515, Boston, MA. Springer US.

Jade Goldstein-Stewart, Ransom Winder, and Roberta Sabin. 2009. Person Identification from Text and Speech Genre Samples. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 336–344, Athens, Greece. Association for Computational Linguistics.

Oren Halvani, Lukas Graner, Roey Regev, and Philipp Marquardt. 2020. An Improved Topic Masking Technique for Authorship Analysis. *arXiv:2005.06605 [cs]*. ArXiv: 2005.06605.

Jaap Kamps, Giannis Tsakonas, Yannis Manolopoulos, Lazaros Iliadis, and Ioannis Karydis. 2017. *Research and Advanced Technology for Digital Libraries: 21st International Conference on Theory and Practice of Digital Libraries, TPDL 2017, Thessaloniki, Greece, September 18-21, 2017, Proceedings*. Springer. Google-Books-ID: it0zDwAAQBAJ.

Bryan Klimt and Yiming Yang. 2004. The Enron Corpus: A New Dataset for Email Classification Research. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, editors, *Machine Learning: ECML 2004*, volume 3201, pages 217–226. Springer Berlin Heidelberg, Berlin, Heidelberg. Series Title: Lecture Notes in Computer Science.

Moshe Koppel and Yaron Winter. 2014. Determining if two documents are written by the same author: Determining If Two Documents Are Written by the

Same Author. *Journal of the Association for Information Science and Technology*, 65(1):178–187.

Xuan Li. Authorship Attribution on the Enron Email Corpus. page 27.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692 [cs]*. ArXiv: 1907.11692.

David Madigan, Alexander Genkin, David D Lewis, Shlomo Argamon, Dmitriy Fradkin, Li Ye, and David D Lewis Consulting. Author Identification on the Large Scale. page 20.

David Madigan, Alexander Genkin, David D. Lewis, and Dmitriy Fradkin. 2005. Bayesian Multinomial Logistic Regression for Author Identification. *AIP Conference Proceedings*, 803(1):509–516. Publisher: American Institute of Physics.

Lukas Muttenthaler, Gordon Lucas, and Janek Amann. Authorship Attribution in Fan-Fictional Texts given variable length Character and Word N-Grams. page 9.

Hendrik Neumann and Martin Schnurrenberger. *E-Mail Authorship Attribution applied to the Extended Enron Authorship Corpus (XEAC)*.

Chen Qian, Tianchang He, and Rao Zhang. Deep Learning based Authorship Identification. page 9.

Sebastian Ruder, Parsa Ghaffari, and John G. Breslin. 2016. Character-level and Multi-channel Convolutional Neural Networks for Large-scale Authorship Attribution. *arXiv:1609.06686 [cs]*. ArXiv: 1609.06686.

Chakaveh Saedi and Mark Dras. 2019. Siamese Networks for Large-Scale Author Identification. *arXiv:1912.10616 [cs]*. ArXiv: 1912.10616 version: 1.

Yunita Sari, Mark Stevenson, and Andreas Vlachos. 2018. Topic or Style? Exploring the Most Useful Features for Authorship Attribution. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 343–353, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Yunita Sari, Andreas Vlachos, and Mark Stevenson. 2017. Continuous N-gram Representations for Authorship Attribution. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 267–273, Valencia, Spain. Association for Computational Linguistics.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James Pennebaker. Effects of Age and Gender on Blogging. page 6.

Yanir Seroussi, Ingrid Zukerman, and Fabian Bohnert. 2014. Authorship Attribution with Topic Models. *Computational Linguistics*, 40(2):269–310.

Prasha Shrestha, Sebastian Sierra, Fabio González, Manuel Montes, Paolo Rosso, and Thamar Solorio. 2017. Convolutional Neural Networks for Authorship Attribution of Short Texts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 669–674, Valencia, Spain. Association for Computational Linguistics.

Juan Soler-Company and Leo Wanner. 2017. On the Relevance of Syntactic and Discourse Features for Author Profiling and Identification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 681–687, Valencia, Spain. Association for Computational Linguistics.

Wei Song, Chen Zhao, and Lizhen Liu. 2019. Multi-Task Learning for Authorship Attribution via Topic Approximation and Competitive Attention. *IEEE Access*, 7:177114–177121. Conference Name: IEEE Access.

Efstathios Stamatatos. 2009. A survey of modern authorship attribution methods. *Journal of the American Society for information Science and Technology*, 60(3):538–556.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2020. How to Fine-Tune BERT for Text Classification? *arXiv:1905.05583 [cs]*. ArXiv: 1905.05583.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *arXiv:1706.03762 [cs]*. ArXiv: 1706.03762.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2020. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv:1910.03771 [cs]*. ArXiv: 1910.03771.

Haoran Wu, Zhiyong Xu, Jianlin Zhang, Wei Yan, and Xiao Ma. 2017. Face recognition based on convolution siamese networks. In *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 1–5.

Min Yang and Kam-Pui Chow. 2014. Authorship Attribution for Forensic Investigation with Thousands of Authors. In *ICT Systems Security and Privacy Protection*, IFIP Advances in Information and Communication Technology, pages 339–350, Berlin, Heidelberg. Springer.

Richong Zhang, Zhiyuan Hu, Hongyu Guo, and Yongyi Mao. 2018. Syntax Encoding with Application in Authorship Attribution. In *Proceedings of the 2018*

Conference on Empirical Methods in Natural Language Processing, pages 2742–2753, Brussels, Belgium. Association for Computational Linguistics.

TREE ADJOINING GRAMMAR BASED "LANGUAGE INDEPENDENT GENERATOR"

Pavan Kurariya, Prashant Chaudhary, Jahnavi Bodhankar, Lenali Singh, Ajai Kumar and Hemant Darbari
Centre for Development of Advanced Computing, Pune, India
pavank@cdac.in, cprashant@cdac.in, jahnavib@cdac.in, lenali@cdac.in, ajai@cdac.in and darbari@cdac.in

Abstract

This paper proposes language independent natural language generator for Tree Adjoining Grammar (TAG) based Machine Translation System. In this model, the TAG based parsing and generation approach considered for the syntactic and semantic analysis of a source language. This model provides an efficient and a systematic way of encapsulating language resources with engineering solution to develop the machine translation System. A TAG based Generator is developed with existing resources using TAG formalism to generate the target language from TAG based parser derivation. The process allows syntactic feature-marking, the Subject-Predicate Agreement marking and multiple synthesized generated outputs in complex and morphological rich language. The challenge in applying such approach is to handle the linguistically diversified features. It is achieved using rule-based translation grammar model to align the source language to corresponding target languages. Nevertheless, this paper also describes the process of lexicalization and explain the state charts, TAG based adjunction and substitution function and the complexity and challenges beneath parsing-generation process.

1 Introduction

Machine Translation is a sub-field of (computational linguistics) under natural language processing (NLP) where computer is act as a human translator. It processes natural language constructs to automate the process of language translation. Every machine translation system requires programs for translation and automated dictionaries and grammars to support translation. Among the various statistical and rule based methodologies, we have researched on Grammar based model for English to Indian language Translation. Lexicalized Tree Adjoining Grammar (LTAG) is a non Chomsky formalism initially proposed in (Joshi et al., 1975) considered to be mildly context grammar that is ideal for any natural language.

The proposed Translation scheme is based on compatible Tags of the source and target languages. A TAG Parser is act as compliers used to analyze the source sentence based on the Tree Adjoining Grammar and construct the source derivation which is the summarization of the state chart processing. TAG Generator is like an interpreter that interprets the source derivation to a target derivation and lexicalizes the target Derivation into Derived Tree Generator which gives us the target Sentence accordingly.

The basic motivation to use this TAG Grammar model is that it is a rule based language independent feature oriented approach. Any complex agglutinative language can be represented using Tags and their unification features. The translation accuracy of this approach depends on the Tree Grammar rather than "bag of corpus". While any statistical approach translation accuracy depends on the corpus, more the corpus size better will be the output. We can still build a robust model for Indian languages with complex verb and noun morphology. Generation of the tree grammar for the source and Target language requires good linguistic knowledge and expertise for providing feature with the grammar.

Related research work of TAG Generator in NLP is presented in Section-2 as Literature survey. Section-3 elaborates more about basic flow of TAG Generator considering the TAG based MTS. Detailed architecture along with internal modules description have also been explained in this section. While Section-4 talks about results and analysis. Conclusion of the paper is done in Section-5.

2 Literature Survey

There has been a fair amount of research in the field of Tree Adjoining Grammar based generation of Machine Translation (MT). Some closely related research work (Joshi et al., 1997) is reported to address the similar work. (Joshi et al., 1997) discussed that TAG may be an appropriate formalism for generation because of their syntactic

Proceedings of the 17th International Conference on Natural Language Processing, pages 138–143
Patna, India, December 18 - 21, 2020.

attributes. The same observation was found in the work of (J. firgen Wedekind et al. 1988), who applies TAGs to the task of generation. Several researchers describe the properties of TAGs for extracting the syntactic processing of a natural language essential for natural-language generation. Although, generation is not a problem in Machine Translation Application as any system that is based on the TAG formalism has to build a generation component by which a TAG can articulate appropriately with semantic information. In this paper, we discuss one such mechanism where source and target grammar are aligned by defining a relation between the rule sets. The recent research in this field can be viewed as an effort to utilize Syntactic and semantic feature during the generation process. As discussed by (J. firgen Wedekind et al. 1988) requires a property of a grammar which specifies that complements be semantically connected to their head while (Stuart M. Shieber et al. 1988) defines a notion of semantic information, a compositional property which guarantees that it can be locally determined whether phrases can contribute to forming an expression with a given meaning. Generation approach that reorder top-down generation (Marc Dymetman et al. 1988) so as to make available information that utilize the top- down recursion also fall into the localizing information. Semantic-head-driven generation (Yves Schabes et al. 1989) uses semantic heads and their complements as a locus of semantic locality.

3 Workflow of TAG Generator

As earlier said, TAG Generation act like interpreters, just like Java virtual machine (JVM) in the Java Programming language which interpret the byte code to machine code, like wise TAG Generator interpret the source derivation to target Language. TAG Generation compresses into three parts: Transfer Model, Derivation generation, Derived tree generator.

As shown in figure 1, Transfer model is a linguistic based model in which source tree is mapped with the target tree i.e. every node of the source tree is mapped with the node of target tree which is called as Link Information. Link Information is an agreement between a source tree and the target tree, illustrate the node mapping between the language pair. Derivation Generation process the Derivation Parser (byte code in java) convert it into intermediate Derivation understand by the Generator intermediate process called the Derived Tree Generator. Derived Tree Generator is the Lexicalization process of the Derivation Generator which gives us the target output and language based feature required for synthesis.

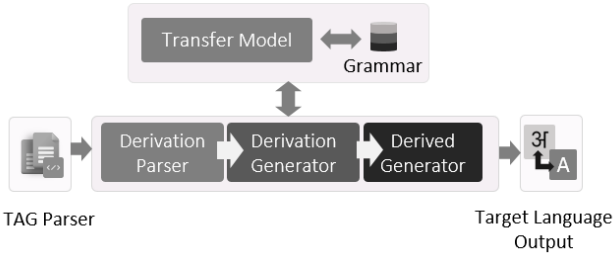


Figure 1: Basic architecture of TAG Generator

3.1 Tree Vector

Tree vector is a very important structure for parser and generator. The tree vector is like a pool for TAG trees, from which the lexicalized trees are spooled up for parsing and generation. The tree vector is a conventional structure implicitly defined as an input to the parser. The tree vector holds maps between trees, tree names and lexicons. There is a string array which holds the segmented sentence with all the words in it. Each word is a key to map, holding the set of tree lexicalized by that word. It also contains a reverse map where a tree is a key to a set of lexicon that uses this particular tree. The most interesting concept of the tree vector is the idea of non repeating trees. The tree vector stores exactly one copy of every tree even if it is lexicalized by many words in the sentence.

3.2 Multithreaded TAG Parser & Derivation

Figure 2, depicts monolithic hybrid parser for Tree Adjunction Grammar (constraint). It is modified multithreaded implementation of 'Early-Type Parsing Algorithm' by Arvind Joshi (Joshi et al. 1997). In this multithreaded parser, every parse requires multithreading and the parser clones a new thread but with a different current state. Multi threaded parser implements the higher RECOGNIZER algorithm. It is an offline recognizer it is designed to identify the first successful parse of an input string. The termination of the thread of the successfully parse is meant to be the end of that iteration. The 'RECOGNIZER' is a non-backtracking algorithm, which instead multi-threads all possibilities at a decision point. The main decisions involves the operation of adjunction and substitution as which tree should be adjuncted or substituted at a given node. The initiation of the parser itself is parallel. Externally the parser starts with 'n' sentence initial

trees, each of which initiates a parse in a different thread. This means that multi-threading occurs at 2 levels, one at the start and then at the parsing level.

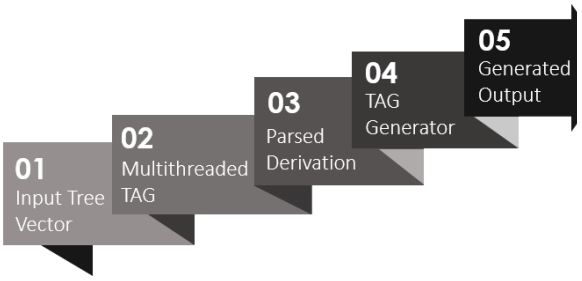


Figure 2: Workflow of TAG Generator

3.3 Transfer Model

It is a linguistic based model in which grammar is written in the tree format for the source and target language. In this model, the source and target grammar are aligned by defining a relation between the rule sets. Similarly of two generative grammar can be mathematically proved to be the similarity in their rule sets and not the language generated. The translation model that we are defining reflects from the proposal in [Abeille, Schabes and Joshi, 1990] is in fact manipulation of multiple similar Grammars. Consider the following illustration where two TAG trees are drawn, similar in nature. The mapping between the nodes is evident here. But there is one problem as the computability of the representation. Through it is complete with respect to the information conveyed; it is computationally redundant. That means the generator will have to compute the actual links from the lexical link given.

Link-info:

1.S_r=2.S_r.~1.NPadjn=2.NPadjn.~1.NP_0=2.NP.~1.VPadjn=2.VPadjn.~1.VP=2.VP.~1.PP=2.PP_1.~1.V=2.ADJ.~

3.4 Target derivation generation

There was an observation during the research on the TAG Generator that how TAG Parser communicates with the TAG Generator. TAG Parser keeps all the parsing information in the states, are stacked in the state chart. TAG Generator doesn't know about the parsing algorithm, state and state chart. To process state chart, you have to know the flow of the parsing so we need one structure which keep the summarization of the state chat and it could be understandable by Generator that interpret in target language.

To summarize the state chart information, we adopted the derivation structure(see figure 3) which is a record of how the elementary trees of a TAG are put together by the operations of substitution and adjoining in order to obtain the derived tree whose yield is the string being parsed. The nodes of the derivation tree are labeled by the names of the elementary trees and the edges are labeled by the addresses of the tree labeling the parent node in which the trees labeling the child nodes are either substituted or adjoined.

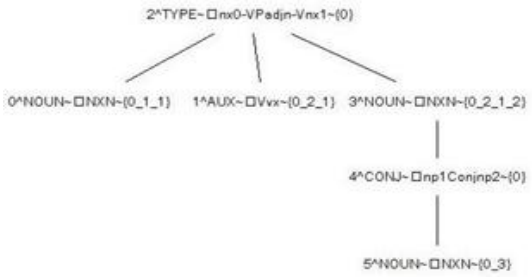


Figure 3: Parser derivation

Generator derivation is the kernel, building the target derivation and tree vector (see figure 4). It also dynamically builds the target tree vector, which is used to clone tree and stitching the clone together to get the final derivation tree. We build the target derivation from the parser derivation. Every node of the parser derivation is mapped with the Target derivation node. During mapping, name of the source elementary tree will be mapped with the target elementary tree, source lexicon will be replace with the target lexicon and Translation model identify in which node of the parent tree, child tree will be adjuncted , it is indicated by gorn number.

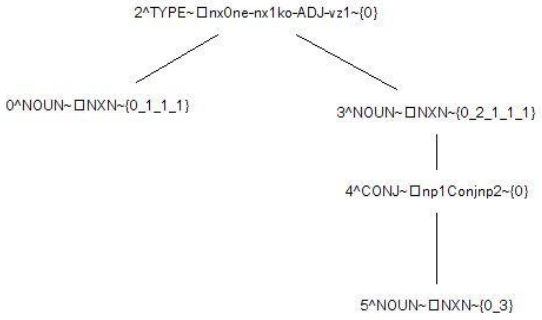


Figure 4: Generated derivation

Major advantages to make derivation is to save memory, don't need to know how the parser

analyze the sentence and easily understandable by generation process. Lexicalization of the generation process could be done by only derivation information, not need of extra information. But there is one drawback, it is very sensitive about grammar, mapping between the nodes in the trees (source and target) should be proper mapped otherwise operation will give the wrong output.

3.5 Parse Tree Builder (Derived Tree Generator)

Another concern in our generation process is when and where will the actual lexicalization of the derived tree happen? The lexicon that a tree is to be lexicalized with is present in the derivation node mapped to that tree. The lexicon that a tree is to be lexicalized with is present in the derivation node mapped to that tree. So this property of the derivation map is used to Preserve and still achieve strong lexicalization of the derived tree that is generated. The target derivation node will preserve the additional information required for smoothing and morph synthesis, so no external map or structure are required to carry them.

The derivation tree summarizes all the translate information stores in the state chart and compresses it to minimal size (see figure 5), easier to manipulate. But to see the actual derivation of how the sentence emerged from the grammar trees, we require the translate tree or more commonly, just, the translate. It is stitching of the entire set of tree used in deriving this particular sentence. But although it is a straight forward problem with a linear time algorithm, it has vast temporal space complexities, when we come to implementation. The TAG grammar derives the sentence in a lexical order and other order should not be assumed. To make it manipulatable in post and pre-order space, we

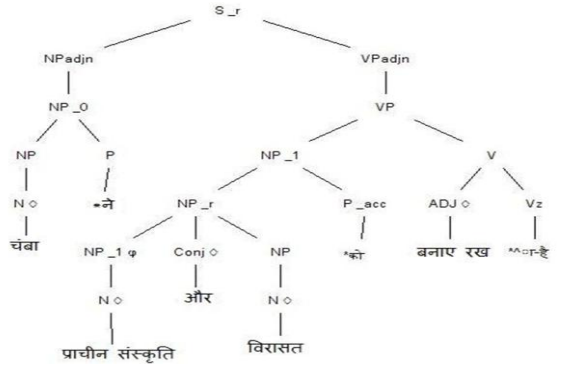


Figure 6: Derived Generation

terms of subject object agreement marker, which gives us the information of the subject, Object and main verb; Possessive case marker in the Indian language like 'Ne', 'Ko', 'Ke-Pass'; multiple synonym generation; multiple output generation based on the context.

TAG Generation approach give us lot of feature in terms of subject object agreement marker, which gives us the information of the subject, Object and main verb; Possessive case marker in the Indian language like 'Ne', 'Ko', 'Ke-Pass'; multiple synonym generation; multiple output generation based on the context.

4 Results and Experiments

Sample sentences covering different type of grammatical structures were generated using TAG Generator in different languages. One of the examples taken from English to Hindi Translation process, here figure 7 and figure8 depict derivation trees created in source and target language. It indicates syntactic relationship between word and how this information has utilized during translation process. Here Tree Vector (see figure 6) is also shown for source sentence.

INPUT TO PP BEAN
The Hawa-Mahal is the most recognizable monument of Jaipur
C-DAC POS OUTPUT
The Hawa-Mahal@@NOUN is @@VERB the most-recognizable-monument @@NOUN of@@PREP Jaipur@@NOUN
APPLIED REDUCTION RULES
@NP+-NP
NP+@VP+NP
@NP+@VP+NP
@NP+-MP
NP+@PP+NP
@NP+-NP
TREE VECTOR
0^ NOUN → [□NXN(Initial)]
1^ TYPE → [□nx0-VPadjn-V(Initial), □nx0-VPadjn-Vnx1-S(Initial), □nx0-VPadjn-Vnx1(Initial), □nx0e-Vpadjn-Vnx1(Initial)]
2^ NOUN → [□NXN(Initia)]
3^ PREP → [□nxPnx(Auxiliary),□vxadjnPnx(Auxiliary)]
4^ NOUN → [□NXN(Initial)]

Figure 5 : Tree Vector

Source Sentence: The Hawa-Mahal is the most recognizable monument of Jaipur.

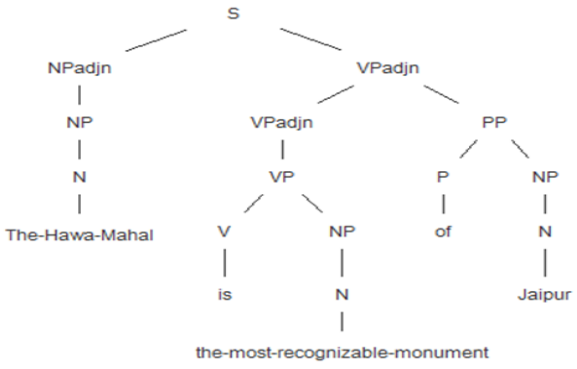


Figure 7: Derived Tress of Parser

Generator Output: हवा महल जयपुर की व्यापक स्तर पर पहचानने योग्य स्मारक है |

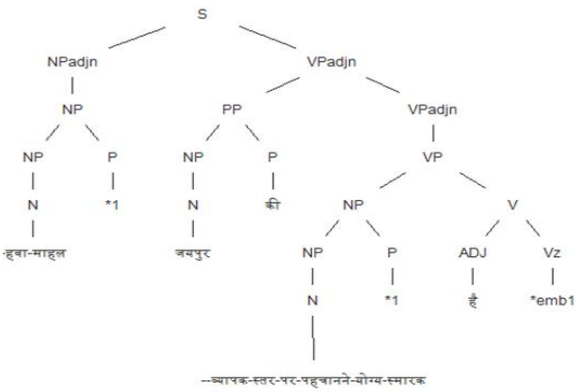


Figure 8: Derived Tress of Generator

4.1 TAG Generator Performance Analysis with PARAM Shavak

Aim of this experiment is to analyze Performance of TAG Generator using multi-core programming on PARAM Shavak*. Virtual Machine has been created by VMware to analyze the performance of Generator. Various test have been carried out to evaluate the Performance of EILMT System with different cores through vnc viewer . During the Test CPU usages and memory Utilizations has been observed.

TAG generator performance experiment on PARAM Shavak with different number of cores.

Data	Number of Cores Time in Second						
	1 core	2 core	4 core	6 core	8 core	12 core	16 core
Sample 1 (217 words)	36	34	32	29	23	20	17
Sample 2 (240 words)	32	21	16	14	12	12	11
Sample 3 (480 words)	58	36	28	26	24	23	21
Sample 4 (960 words)	110	74	52	49	46	42	40

Table 1: Performance experiment table

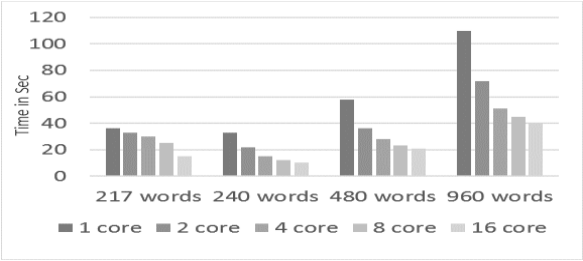


Figure 10: TAG generator performance experiment on PARAM Shavak

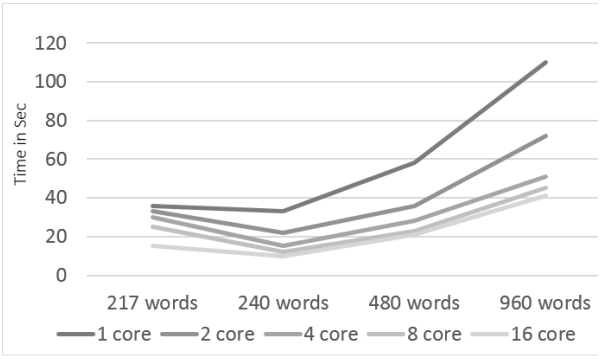


Figure 11: TAG generator performance experiment on PARAM Shavak

Above graph shows as we increase sample size, execution time decreases with each added cores during experiment while CPU usage increased 90-99 % and Memory utilization has been observed (2.1 to 2.8 GB) during test run

*PARAM Shavak: Supercomputer in a Box solution, aims to provide computational resource with advanced technologies to perform the high-end computations on a larger scale for the scientific, engineering and academic programmers. PARAM Shavak is a ready-to-use affordable supercomputer pre-loaded with all the required system software and applications from selected scientific and engineering domains.

5 Conclusion

In this paper, we have discussed some of the major tasks involved in the development of TAG generator using TAG formalism for translation from English to Hindi Language. All the examples illustrated above are taken from the output generated by the machine translation system developed by us. The effectiveness of this approach is tested by experiments on sample corpus abstract taken from existing resources. We have implemented TAG generator as a part of research in Machine Translation system based on Tree Adjoining Grammar. Firstly, we carried out experiments on web application (integrated MT System using TAG based Parser and Generator) running on Windows and Linux platform to provide a baseline. We have demonstrated that Parser and Generator are two core components in Machine Translation System. The system can handle simple and complex sentences with considerably good accuracy rate.

References

Joshi, A. K., Levy, L., and Takahashi, M. (1975). *Tree Adjunct Grammars*. Journal of Computer and System Sciences.

Joshi, A. K and Yves Schabes. 1997. *Tree Adjoining Grammars*. In Handbook of Formal Languages, volume 3, pages 69–123. Springer-Verlag, Berlin.

Nederhof, M.-J. (1998). *an alternative LR algorithm for TAGs*. In Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 16th International Conference on Computational Linguistics, Montreal, Canada.

Khalil Sima'an. 2000. *Tree-gram parsing: Lexical dependencies and structural relations*. In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics, Hong Kong, China

Prolo, C. A. (Feb., 2002b). *LR parsing for Tree Adjoining Grammars and its application to corpus-based natural language parsing*. Ph.D. Dissertation Proposal, Department of Computer and Information Science, University of Pennsylvania.

Karin Harbusch and Jens Woch. 2000d. *Reuse of plan–based knowledge sources in a uniform tag–based generation system*. Under submission, see: http://www.uni-koblenz.de/~harbusch/plantotag.ps.

Schabes, Y. and Vijay-Shanker, K. (1990). *Deterministic left to right parsing of tree adjoining languages*. In Proceedings of 28th Annual Meeting of the Association for Computational Linguistics, pages 276–283, Pittsburgh, Pennsylvania, USA.

Tomita, M. (1985). *Efficient Parsing for Natural Language.* Kluwer Academic Publishers, Boston, MA, USA.

Nicolas Nicolov. 1998. *Memoisation in sentence generation with lexicalized grammars*. In Abeill´e et al.(Abeill´e et al., 1998), pages 124–127.

XTAG Research Group, T. (1998). *A Lexicalized Tree Adjoining Grammar for English.* Technical Report IRCS 98-18, University of Pennsylvania.

David D. McDonald and James D. Pustejovsky. *TAGs as a grammatical formalism for generation.* In Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics, pages 94-103, University

J. firgen Wedekind. *Generation as structure driven derivation*. In Proceedings of the 12th International Conference on Computational Linguistics, pages 732- 737, Budapest, Hungary, 1988.

Stuart M. Shieber. *A uniform architecture for parsing and generation*. In Proceedings of the 12th International Conference on Computational Linguistics, pages 614-619, Karl Marx University of Economics, Budapest, Hungary, 22-27 August 1988.

David D. McDonald and James D. Pustejovsky. *TAGs as a grammatical formalism for generation*. In Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics, pages 94-103, University

J. firgen Wedekind. *Generation as structure driven derivation*. In Proceedings of the 12th International Conference on Computational Linguistics, pages 732-737, Budapest, Hungary, 1988.

Stuart M. Shieber. *A uniform architecture for parsing and generation*. In Proceedings of the 12th International Conference on Computational Linguistics, pages 614-619, Karl Marx University of Economics, Budapest, Hungary, 22-27 August 1988.

Marc Dymetman and Pierre Isabelle. *Reversible logic grammars for machine translation*. In Proceedings of the Second International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages, Pittsburgh, Pennsylvania, 1988. Carnegie-Mellon University.

Yves Schabes and Aravind K. Joshi. *The relevance of lexicalization to parsing*. In Proceedings of the International Workshop on Parsing Technologies, pages 339-349, Pittsburgh, Pennsylvania, 28-31 August 1989. Carnegie-Mellon University.

Exploration of Cross-lingual Summarization for Kannada-English Language Pair

Vinayaka R Kamath Rachana Aithal K R K Vennela Mamatha HR
Department of Computer Science & Engineering
PES University, India
vinayakarkamath@pesu.pes.edu
{rachanaaithal88, kvennela1998}@gmail.com
mamathahr@pes.edu

Abstract

Cross-lingual summarization(CLS) is a process in which given a document in source language aims at generating summary in a different, destination language. Low resource languages like Kannada greatly benefit from such systems because they help in delivering a concise representation of the same information in a different popular language. We propose a novel dataset generation pipeline and a first of its kind dataset that will aid in CLS for both English-Kannada and Kannada-English pair. This work is also an attempt to inspect the existing systems and extend them to the Kannada-English language pair using our dataset.

1 Introduction

With the advancement in technology, language should not be a barrier to gain knowledge when everyone has access to the same information. The need for an intelligent system that understands and analyzes text in low resource languages while delivering concise representation of the text in a well known language without losing out on any information is paramount. Cross-lingual summarization systems fit these requirements perfectly. For a given document in the source language, the primary objective of the system is to generate meaningful summary in the target language (different from source language) without discarding any crucial information. These systems extend resources available in low resources languages like Kannada to everyone who can understand a well known language such as English.

Monolingual summarization is extensively studied due to the availability of resources while cross-lingual summarization systems are relatively unpopular due to the lack of training corpus. To tackle this, we present a one of it's kind dataset for training a CLS system for Kannada-English language pair along with our experimentation. A robust pipeline for the generation of dataset is designed using round trip translation strategy from (Zhu et al., 2019) and a back translation strategy proposed by (Duan et al., 2019) using the Newsroom dataset from (Grusky et al., 2018) as our primary backbone. We have successfully extended several methods from (Wan, 2011), (Jhaveri et al., 2019) and some baselines from (Shen et al., 2018) to Kannada-English language pair using our dataset. Section 2 describes the dataset generation pipeline and the experimentation carried out. The results and inferences are discussed in section 3 while the conclusions are briefed in section 4.

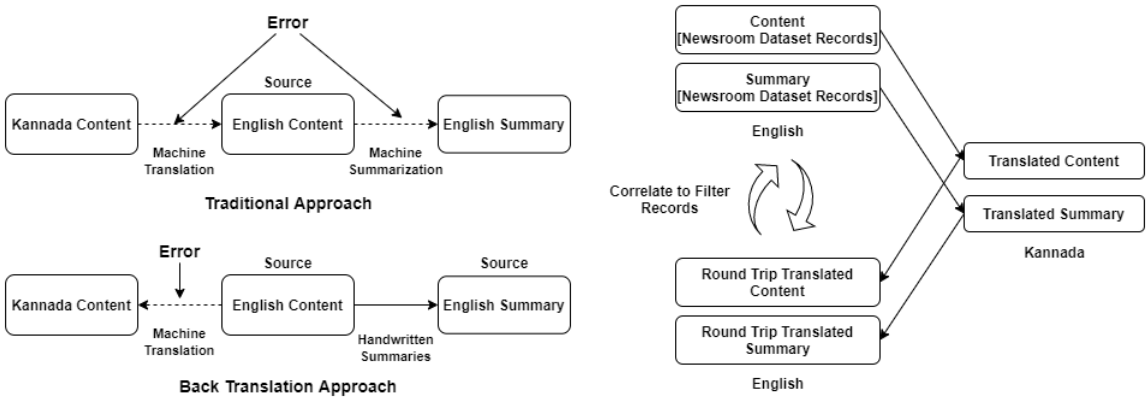
2 Experimentation

2.1 Dataset Construction

This section describes the methods used to construct the very first Kannada-English summarization dataset. The absence of a CLS corpus for Kannada-English language pair is a significant hindrance. To overcome this, we propose a novel pipeline and a newly constructed high quality dataset that will aid in CLS for Kannada-English language pair.

The pipelined process of translation followed by summarization of the content in source language introduces a lot of noise in the generated data. Moreover, the process thoroughly utilises a third party "automated machine translation system". To elevate the quality, back translation strategy was implemented using the English content as pivot as specified in Fig.1a. This ensured that there was little dependency on third party systems and minimal noise in the generated data. However, this method can only be successful if a good quality source dataset is used. Cornell newsroom is one such backbone that we used while building our dataset. The Cornell Newsroom dataset(Grusky et al., 2018) is a large monolingual dataset for training and evaluating summarization systems. It con-

Proceedings of the 17th International Conference on Natural Language Processing, pages 144–148
Patna, India, December 18 - 21, 2020.



(a) Translation Approach Comparison

(b) Round Trip Translation Procedure

Figure 1: Illustrations of methods used to create the dataset

tains 1.3 million articles and summaries written by authors and editors in the newsrooms of 38 major publications.

Filtering noisy outliers is crucial to ensure highest quality possible. The Round Trip Translation(RTT) mechanism described in (Zhu et al., 2019) is used to achieve this. The RTT strategy is used to acquire high-quality large-scale cross lingual summarization dataset from existing large-scale monolingual dataset(Fig.1b). It can be observed that the current monolingual translators are not very proficient. This may result in addition of considerable amount of noise in the dataset if it is constructed by direct translation. Therefore, to improve the quality of parallel corpus the RTT mechanism is employed. This involves calculation rouge score between the reference content and hypothesis content which we obtained after round trip translation. This is proceeded by the filtering of the dirty samples by choosing samples above a threshold value. This ensures that the dataset is reliable and efficient. The threshold is decided by manually sampling the records of the dataset at different values of the rouge scores and visually inspecting the quality of the record. The records are assessed for sentence completion, preservation of semantic meaning and external noise. A suitable threshold is which acts as a cutoff value for noisy records. A heuristic threshold is chosen by trading off the number of records in the dataset to the peak quality of the records. The same procedure is followed to generate summaries(including RTT summaries) as well, this is to make sure that the dataset maps both English content to it's corresponding Kannada summary as well as the Kannada content to it's English summary. This ensures that the dataset is capable of aiding in training both Kannada to English as well as English to Kannada CLS systems.

As a result of our proposed system, we constructed a high quality dataset of 23,113 records that supports interchangeability of source-target languages. The whole dataset is made publicly available along with the rouge scores to filter the records as per the requirements of the application.

2.2 Cross Lingual Summarization Systems

Extending the current state-of-the-art CLS methodologies to the regional language of Kannada can accelerate the process of designing a robust system that can be used to generate good quality summaries for the content in Kannada. This section briefly describes the methods extended to Kannada as illustrated in Figure 2.

2.2.1 Baselines

Early translation systems are used as baselines. During early translation, the process of translation and summarization are stacked in that order to form a simple cross-lingual summarizer. This system relies on the good quality translators and summarizers available for a high resource language like English. In our work, four mono-lingual summarization algorithms namely LSA, LexRank, Luhn and TextRank are used to extract summaries from the translated documents.

Latent Semantic Analysis (LSA) (Steinberger and Jezek, 2004) is an algebraic-statistical method that extracts hidden semantic structure of words and sentences. It relies on Term Frequency-Inverse Document Frequency (TF-IDF) and Singular Value Decomposition (SVD) to achieve summarization

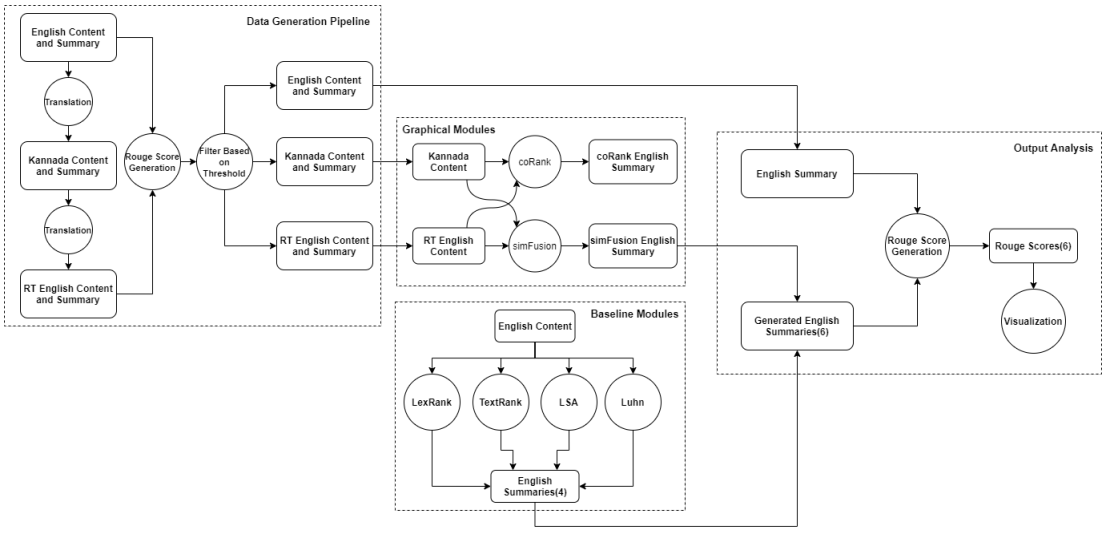


Figure 2: Schematic of the complete experimentation stack

of the text. The matrix constructed using TF-IDF is subjected to decomposition using SVD, there after the topic method is used to extract concepts and sub-concepts to select important sentences. These selected sentences are ranked and presented as a concise summary. LexRank (Erkan and Radev, 2004) is an unsupervised approach to text summarization based on graph-based centrality scoring of sentences. The algorithm recommends sentences that are very similar to the others in the document, thus curbing redundant information in the output. Luhn algorithm (Torres-Moreno, 2014) takes a naive approach based on TF-IDF that concentrates on the window size of non-important words between words of high importance. The summary is generated by assigning weights to the words and recommending sentences with maximum weight values. TextRank (Mihalcea and Tarau, 2004) is a graph based sentence ranking algorithm that uses PageRank algorithm to attain convergence. The algorithm is very similar to LexRank but uses simpler methods to accelerate the computation.

2.2.2 Extended Models

This section describes the sub-modular function maximization based summarization algorithms that were adopted and used for cross-lingual settings. coRank(Jhaveri et al., 2019) and simFusion(Xi et al., 2005) were extended to Kannada-English language pair and a thorough analysis of the results was performed to understand the semantic suitability of the techniques.

coRank method leverages both the Kannada lexicon information and the English-side information in a co-ranking way. The source Kannada sentences and the translated English sentences are simultaneously ranked in a unified graph-based algorithm. The saliency of each Kannada sentence relies not only on the Kannada sentences linked with it, but also the corresponding English sentences associated with it and the same holds true for English sentences. simFusion algorithm uses the Kannada side information for English sentence ranking in the graph-based framework. The similarity value between two English sentences is computed by linearly fusing the similarity value between the corresponding two Kannada sentences with its very own. The graph is constructed using the similarity in both the source and the target languages. In both the methods, the sentences with the highest saliency scores are compiled together to give the summary in the target language.

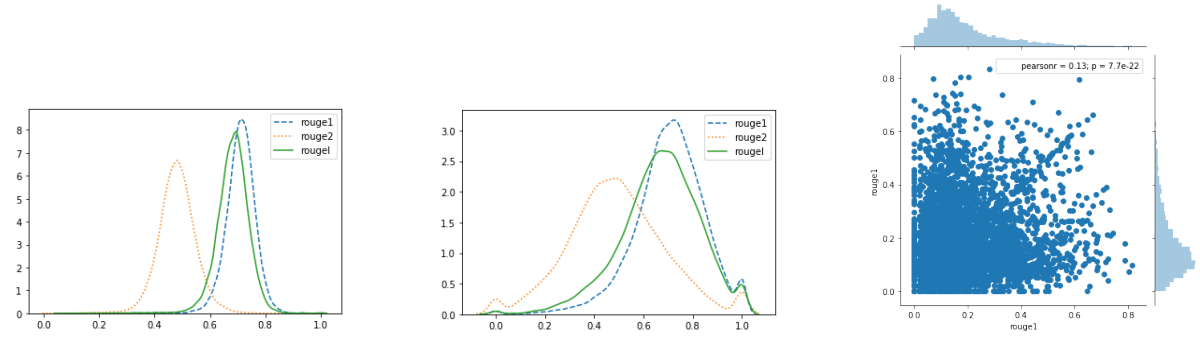
3 Results

The rouge scores of the round trip translated corpus with the records from the newsroom dataset were recorded. This was done in order to inspect the records and set an appropriate threshold to filter out the noisy records. Removing these unwanted records helped in increasing the quality of the dataset. The same observation was recorded for the summaries as well. This enabled the dataset to be more flexible with the interchangeability of the source and destination languages.

The threshold was decided after inspecting the distribution of the rouge scores between round trip translated data and the original summary. Our ex-

	Algorithm	Rouge 1	Rouge 2	Rouge l
CLS Baselines	LSA	17.911	5.611	12.4041
	LexRank	19.6429	6.2439	14.2806
	Luhn	19.1805	6.2301	13.8486
	TextRank	19.3113	6.2806	13.5528
CLS Graphical	coRank	18.7916	6.3136	13.1178
	simFusion	18.3779	6.0298	12.8286
Popular Summarization Systems on Newsroom	Seq2Seq + Attention (Rush et al., 2015)	5.99	0.37	5.41
	Fast-RL (Keneshloo et al., 2019)	21.93	9.37	19.61
	ExtConSumm (Mendes et al., 2019)	39.40	27.80	36.20
	Modified P-G (Shi et al., 2019)	39.91	28.38	36.87

Table 1: Results from Experimentation.



(a) Distribution of Content Rouge Scores

(b) Distribution of Summary Rouge Scores

(c) Correlation between coRank and simFusion Results

Figure 3: Analysis of the results

perimentation of manual inspection by evenly sampling 30% of the records by 3 volunteers yielded 0.24 with variance of 0.03, the next stages of the experiments were carried forward with this threshold in mind. Figures 3a and 3b helped in filtering the records by providing the overview of the distribution. The attempt to check for redundancy in the outputs between the results exhibited by the simFusion and coRank algorithms were done. Although both of them follow a graphical approach, Figure 3c proved that the information captured by these algorithms were quite different. There was very little correlation between the scores of two algorithms for the same set of records, this implies that an ensemble of both these models can provide maximum outreach.

The rouge scores achieved from the experimentation with the different CLS methods are depicted in Table 1. The set of CLS experiments were conducted with Kannada as the source language and English as the destination language for the task at hand. These results are compared among themselves as well as other popular summarization systems on Newsroom dataset, since a cross-lingual summarization system for Kannada-English pair does not exist yet.

4 Conclusion

In this paper, we proposed a first of its kind dataset that consists of content-summary mappings for the Kannada-English language pair. Since Kannada is a low resource language, the dataset can aid for further experimentation on cross lingual applications. The newly designed dataset generation pipeline has also been proven to generate high quality records considering the CLS methods that has been successfully extended to Kannada-English language pair using the dataset. Table 1 shows that the results from our experiments are comparable to that of those that have used the same corpus for designing systems. These results can act as a solid foundation for further exploration. The results from the Table 1 also act as a proof of correctness for the experimental setup. We believe that the first set

of CLS experiments for Kannada presented in this paper has set reasonable benchmarks with adequate resources to carry forward computational linguistic experiments for a low resource language. We intend to design/implement systems that use the translated content along with the source content to perform better at CLS tasks centered around Kannada as a part of our future work.

References

Xiangyu Duan, Mingming Yin, Min Zhang, Boxing Chen, and Weihua Luo. 2019. Zero-shot cross-lingual abstractive sentence summarization through teaching generation and attention. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3162–3172.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 708–719, New Orleans, Louisiana. Association for Computational Linguistics.

Nisarg Jhaveri, Manish Gupta, and Vasudeva Varma. 2019. clstk: The cross-lingual summarization toolkit. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 766–769.

Yaser Keneshloo, Naren Ramakrishnan, and Chandan K Reddy. 2019. Deep transfer reinforcement learning for text summarization. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 675–683. SIAM.

Afonso Mendes, Shashi Narayan, Sebastião Miranda, Zita Marinho, André FT Martins, and Shay B Cohen. 2019. Jointly extracting and compressing documents with summary state representations. *arXiv preprint arXiv:1904.02020*.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Shi-qi Shen, Yun Chen, Cheng Yang, Zhi-yuan Liu, Mao-song Sun, et al. 2018. Zero-shot cross-lingual neural headline generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(12):2319–2327.

Tian Shi, Ping Wang, and Chandan K. Reddy. 2019. LeafNATS: An open-source toolkit and live demo system for neural abstractive text summarization. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 66–71, Minneapolis, Minnesota. Association for Computational Linguistics.

Josef Steinberger and Karel Jezek. 2004. Using latent semantic analysis in text summarization and summary evaluation. *Proc. ISIM*, 4:93–100.

Juan-Manuel Torres-Moreno. 2014. *Automatic text summarization*. John Wiley & Sons.

Xiaojun Wan. 2011. Using bilingual information for cross-language document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1546–1555. Association for Computational Linguistics.

Wensi Xi, Edward A. Fox, Weiguo Fan, Benyu Zhang, Zheng Chen, Jun Yan, and Dong Zhuang. 2005. Simfusion: Measuring similarity using unified relationship matrix. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, page 130–137, New York, NY, USA. Association for Computing Machinery.

Junnan Zhu, Qian Wang, Yining Wang, Yu Zhou, Jiajun Zhang, Shaonan Wang, and Chengqing Zong. 2019. Ncls: Neural cross-lingual summarization. *arXiv preprint arXiv:1909.00156*.

Hater-O-Genius Aggression Classification using Capsule Networks

Parth Patwa[1] **Srinivas PYKL**[1] **Amitava Das**[2]
Prerana Mukherjee[1] **Viswanath Pulabaigari**[1]
[1]Indian Institute of Information Technology Sri City, India [2]Wipro AI Labs, India
[1]{parthprasad.p17, srinivas.p, prerana.m, viswanath.p}@iiits.in
[2]amitava.das2@wipro.com

Abstract

Contending hate speech in social media is one of the most challenging social problems of our time. There are various types of anti-social behavior in social media. Foremost of them is aggressive behavior, which is causing many social issues such as affecting the social lives and mental health of social media users. In this paper, we propose an end-to-end ensemble-based architecture to automatically identify and classify aggressive tweets. Tweets are classified into three categories - Covertly Aggressive, Overtly Aggressive, and Non-Aggressive. The proposed architecture is an ensemble of smaller subnetworks that are able to characterize the feature embeddings effectively. We demonstrate qualitatively that each of the smaller subnetworks is able to learn unique features. Our best model is an ensemble of Capsule Networks and results in a 65.2% F1 score on the Facebook test set, which results in a performance gain of 0.95% over the TRAC-2018 winners. The code and the model weights are publicly available at https://github.com/parthpatwa/Hater-O-Genius-Aggression-Classification-using-Capsule-Networks.

1 Introduction

Even though social media offers several benefits to people, it has caused some negative effects due to the misuse of freedom of speech by a few people.

Aggression is a behavior that is intended to harm other individuals who do not wish to be harmed (O'Neal, 1994). Aggressive words are commonly used to inflict mental pain on the victim by showing covert aggression, overt aggression or by using offensive language (Davidson et al., 2017).

The process of manually weeding out aggressive tweets from social media is expensive and indefinitely slow. So, there is a growing need to build and analyze automatic aggression classifiers.

In this paper, we propose an architecture that is an ensemble of multiple subnetworks to identify aggressive tweets, where each subnetwork learns unique features. We explore different word embeddings for dense representation (Mikolov et al., 2013), deep learning (CNN, LSTM), and Capsule Networks (Sabour et al., 2017). Our best model (figure 1) uses Capsule Network, and gives a 65.20% F1 score, which is a 0.95% improvement over the model proposed by Aroyehun and Gelbukh (2018). We also release the code and the model weights.

2 Related Work

The challenge of tackling antisocial behavior like abuse, hate speech, and aggression on social media has recently received much attention. Researchers like Nobata et al. (2016) tried detecting abusive language by using Machine Learning and linguistic features. Other researchers like Badjatiya et al. (2017) used CNNs and LSTMs, along with gradient boosting, to detect hate speech.

The TRAC-2018 shared task (Kumar et al., 2018a), aimed to detect aggression, was won by Aroyehun and Gelbukh (2018), who used deep learning, data augmentation, and pseudo labeling to get a 64.25% F1 score. Another team Risch and Krestel (2018), used deep learning along with data augmentation and hand-picked features to detect aggression. However, in order to develop an end-to-end automated system, one cannot use hand-picked features as they may vary from dataset to dataset. Srivastava et al. (2018) experimented with capsulenets for detecting aggression and achieved a 63.43% F1 score. Our work differs from theirs as we experiment with architectures (Fig. 1) that are an ensemble of multiple subnetworks. Recently,

Proceedings of the 17th International Conference on Natural Language Processing, pages 149–154
Patna, India, December 18 - 21, 2020.

Khandelwal and Kumar (2020) used pooled biLSTM and NLP features to achieve 67.7% F1 score on the TRAC-2018 Facebook data.

The TRAC-2020 shared task (Kumar et al., 2020) released a data set (Bhattacharja, 2010) of aggression and misogyny in Hindi, English and Bengali posts. Risch and Krestel (2020) tried an ensemble of BERT to achieve the best performance on most tasks. Safi Samghabadi et al. (2020) used BERT in a multi-task manner to solve the task, whereas Kumari and Singh (2020) used LSTM and CNNs.

3 Dataset

To identify the type of aggression, we use the English train dataset, and the Facebook (fb) test dataset provided by the 2018 TRAC shared task (Kumar et al., 2018a). The data collection and annotation method is described in Kumar et al. (2018b). The training data is combined with the augmented data provided by Risch and Krestel (2018). The final distribution is given in table 1. The data has English-Hindi code-mixed tweets, which are annotated with one of three labels:

- **Covertly Aggressive (CAG)**: Behavior that seeks to indirectly harm the victim by using satire and sarcasm (Kumar et al., 2018b). E.g., *"Irony is your display picture at one end you are happy seeing some one innocent dying and at other end you are praying to not kill an innocent"*
- **Overtly Aggressive (OAG)**: Direct and explicit form of aggression which includes derogatory comparison, verbal attack or abusive words towards a group or an individual (Roy et al., 2018). E.g., *"Shame on you assholes showing some other video and making it a fake news u chooths i hope each one you at *** news will rot in hell"*
- **NAG**: Texts which are not aggressive. E.g., *"hope car occupants are safe and unharmed."*

We observe that the dataset contains some tweets which have improbable annotations. For example, the tweet *"Mr. Sun you are wrong, Pakistan produces one thing that is ' terrorists' and through CPEC Pak will increase the supply of this product throughout world. Wait you will feel the touch of their product in your Muslim dominated province."* is labeled as NAG; *"#salute you my friend"* is labeled as OAG. To have a fair comparison with the results of previous works, we don't do anything to address this. The dataset is imbalanced with maximum tweets labeled as NAG.

Table 1: Data distribution

Class	Train	Test
Covertly Aggressive	14,187	144
Overtly Aggressive	9,137	142
Non-Aggressive	16,188	630
Total	39,512	916

4 Preprocessing and Embeddings

The tweets are first converted to lower case. Next, we remove digits, special characters, emojis, urls, and stop words. We restrict the continuous repetition of the same character in a word to 2 (e.g. 'suuuuuuper' is converted to 'suuper'). Each tweet is tokenized and converted into a sequence of integers. The maximum sequence length is restricted to 150. To have dense representation of tokens, the following word embedding features are used:

- **Glove++:** Given the word, we first check whether it is present in Glove pre-trained 6b 100d embeddings, and use the embedding if it exists. For Out-Of-Vocabulary words, we use the word vectors that we train on the entire data using the Gensim library.
- **Aggression Embeddings:** To have distinguishing features to separate aggressive tweets from non-aggressive tweets, we create aggression word embeddings. We take all the tweets classified as OAG and CAG and train word vectors on them.
- **Char Trigram:** To get sub-word information, we create character trigram embeddings.

5 Proposed Architecture

We propose an architecture that combines features that are learned from an ensemble of subnetworks and leverages the feature representation to classify aggression. All models optimize the categorical crossentropy loss function using adam optimizer. All the dense layers, except the final layer, have ReLu activation. All the CNN layers are followed by dropout = 0.5. Every model is an ensemble of

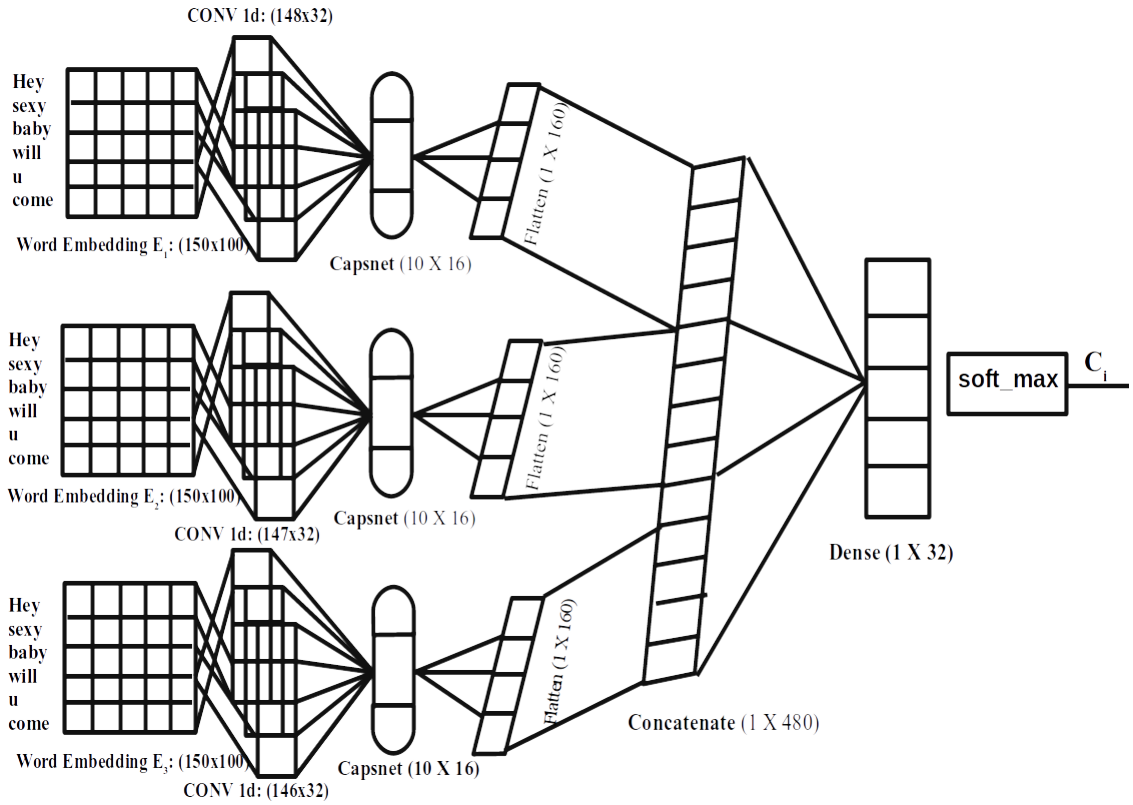


Figure 1: Architecture of CN1 model

smaller subnetworks. Each subnetwork (SN) has the following configuration:

- **Embedding layer** - Each token in the input sequence is represented by its word vector. Word embeddings help to capture the meaning of the word.
- **Convolutional layer** - A convolutional layer, having reLu activation function, to extract spatial features.
- **Max-pooling layer** of size 2 or 3 in case of Deep Learning models.
- **Capsule layer** to better preserve spatial information, in case of Capsulenet models.

Each SN of the model uses a different configuration for the CNN layer or embedding. Therefore each SN learns different information and generates different features. The output of each SN is flattened and merged and is passed as input to dense layers. The last dense layer has three neurons and a softmax activation function, which gives a probability score to each of the three classes, and the one with the highest score is the predicted class.

5.1 Deep Learning (DL) Models

The following are the DL baselines:

DL1: It is an ensemble of three subnetworks. All three SNs use Glove++ embeddings for the embedding layer. The CNN layers in each SN have kernel sizes 3,5 and 7, respectively.

DL2: It is an ensemble of 9 SNs. Each max-pooling layer is followed by a biLSTM layer, having 200 units, to capture long term dependencies. SN 1-3 use Glove++ embeddings. SN 4-6 use Aggression embeddings. SN 7-9 use Character-level trigram embeddings. CNN layer in SN 1,4,7 has kernel size = 3, in SN 2,5,8 has kernel size = 5 and in SN 3,6,9 has kernel size = 7.

5.2 Capsule Network (CN) Models

The main difference between CN models and DL models is that the CN models use a capsule layer instead of max-pooling layer. The capsule layer has 10 capsules of 16 dimension each. Max-pooling reduces computational complexity but leads to the loss of spatial information.

Capsules are a group of neurons that are represented as vectors. The orientation of the feature vector is preserved in capsules. They use a function called squashing for non-linearity. Dynamic

DL models		CN models	
DL1	57.17%	CN1	**65.20%**
DL2	60.34%	CN2	62.70%

Table 2: Weighted F1 scores of DL and CN models

Routing is used to route the feature vector of the lower-level capsule to the appropriate next level capsule (Sabour et al., 2017). Dynamic Routing is based on a coupling coefficient that measures the similarity between vectors that predict the upper capsule and the lower capsule and learns which lower capsule should be directed to which upper capsule (Kim et al., 2018). Through this process, capsule layers preserve spatial information, learn semantic representation, and ignore words that are insignificant.

CN1: The architecture is shown in figure 1. It is an ensemble of 3 subnetworks. Each SN uses Glove++ embeddings, and the CNN layers have kernel size = 3,4 and 5, respectively.

CN2: Like CN1, but there is an additional biLSTM layer, having 300 units, after the capsule layer.

6 Results and Discussion

From table 2, we see that the CN models perform better than DL models. Both the CN models are comparable to the models proposed by Srivastava et al. (2018). This validates the usefulness of capsule networks for aggression detection. CN1 gives the best results and is better than the best model proposed by Aroyehun and Gelbukh (2018). DL2 works better than DL1, as it captures more information. The performance drops from CN1 to CN2, despite CN2 having an additional biLSTM layer. This shows that a more complex model is not necessarily better, which is in agreement with the observations of Aroyehun and Gelbukh (2018). This could be due to over-fitting.

Figures 3, 4 and 5 are t-SNE (van der Maaten and Hinton, 2008) graphs, which depict the output of SN1-3 of CN1, respectively. We visualize the feature embeddings in all the SNs, and we observe that each SN is able to characterize the features distinctly due to the variability in the network configurations. When all the SNs are combined in an ensemble network, the feature representation is further improved. The inter-class variability is predominant, as can be validated in Fig. 6. This can be attributed to the fact that all 3 SNs have complimentary feature representations.

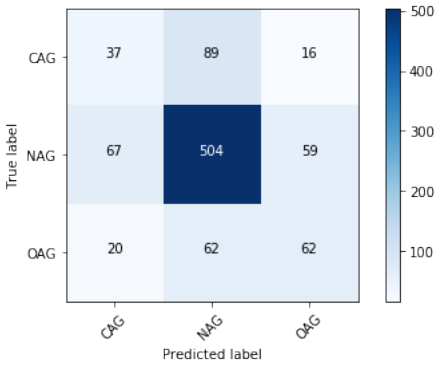


Figure 2: Confusion matrix of CN1 model

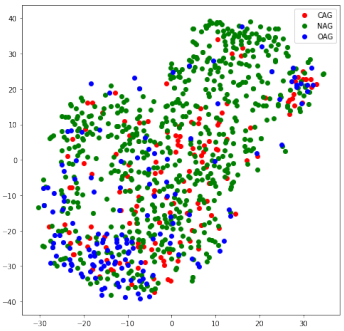


Figure 3: Flatten vector of subnetwork1

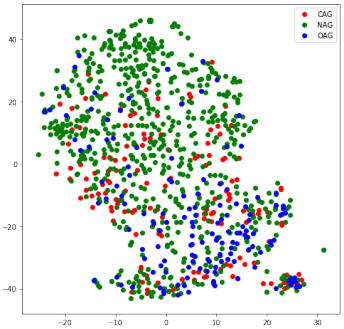


Figure 4: Flatten vector of subnetwork2

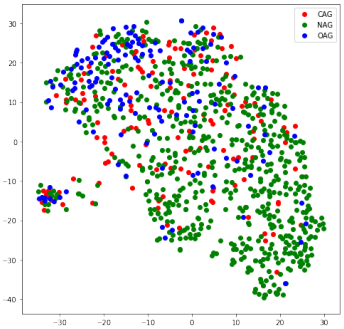


Figure 5: Flatten vector of subnetwork3

As observed from the confusion matrix of CN1 model (Fig. 2), NAG is the easiest to detect.

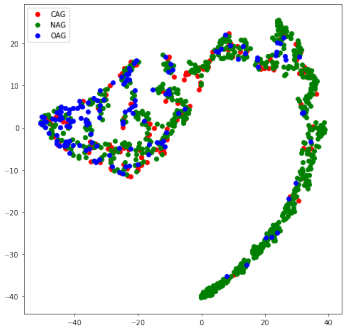


Figure 6: Performance of Output Layer

It is because most of the tweets in the data are NAG. The performance is better on OAG than on CAG, despite there being more training examples of CAG as OAG is more explicit and hence easier to identify, as opposed to the more indirect CAG (Davidson et al., 2017). CAG, because of its covert nature is the most difficult to classify. The confusion of CAG can also be observed in figure 6, where CAG is overlapping with NAG and OAG.

The confusion can also be seen by analyzing some **CAG** tweets **predicted** as **NAG**:
"*Hundreds of people were killed by your friends in Bombay, where were you at that time.*"
"*What's next? Soon we will be told to have a bullock cart and give up cars? Or live in a shed using candles?*"
"*Chit fund operators n loan sharks r more honest*"

7 Conclusion and Future Work

We perform experiments to identify aggressive tweets by applying DL and Capsule Networks on preprocessed data. We show that capsulenets are efficient for aggression detection. We use an ensemble-based model and qualitatively show that each subnetwork learns unique features which help in classification. Our best model uses capsulenets and results in a 65.20% f1 score, which is an improvement over most of the existing solutions.

In the future, we would like to explore other capsulenet architectures using different routing algorithms. A more in-depth analysis of CAG tweets could improve the performance on them.

8 Acknowledgement

We thank the anonymous reviewers for their constructive feedback. We also thank Rohan Sukumaran and Harshit Singh for fruitful discussions and proofreading.

References

Segun Taofeek Aroyehun and Alexander Gelbukh. 2018. Aggression detection in social media: Using deep neural networks, data augmentation, and pseudo labeling. In *TRAC - 2018*, pages 90–97.

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, pages 759–760, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Shishir Bhattacharja. 2010. Bengali verbs: a case of code-mixing in Bengali. In *Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation*, pages 75–84, Sendai, Japan.

Maral Dadvar, Dolf Trieschnigg, Roeland Ordelman, and Franciska de Jong. 2013. Improving cyberbullying detection with user context. In *Advances in Information Retrieval*, pages 693–696.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. ICWSM '17, pages 512–515.

Anant Khandelwal and Niraj Kumar. 2020. A unified system for aggression identification in english code-mixed and uni-lingual texts. In *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*, CoDS COMAD 2020, page 55–64, New York, NY, USA. Association for Computing Machinery.

Jaeyoung Kim, Sion Jang, Sungchul Choi, and Eunjeong L. Park. 2018. Text classification using capsules. *CoRR*, abs/1808.03976.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018a. Benchmarking aggression identification in social media. In *TRAC-2018*, pages 1–11.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2020. Evaluating aggression identification in social media. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 1–5, Marseille, France. European Language Resources Association (ELRA).

Ritesh Kumar, Aishwarya N. Reganti, Akshit Bhatia, and Tushar Maheshwari. 2018b. Aggression-annotated corpus of hindi-english code-mixed data. *CoRR*, abs/1803.09402.

Kirti Kumari and Jyoti Prakash Singh. 2020. AI_ML_NIT_Patna @ TRAC - 2: Deep learning approach for multi-lingual aggression identification. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 113–119, Marseille, France. European Language Resources Association (ELRA).

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 145–153, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Edgar C. O'Neal. 1994. Human aggression, second edition, edited by robert a. baron and deborah r. richardson. new york, plenum, 1994, xx + 419 pp. *Aggressive Behavior*, 20(6):461–463.

Julian Risch and Ralf Krestel. 2018. Aggression identification using deep learning and data augmentation. In *TRAC-2018*, pages 150–158.

Julian Risch and Ralf Krestel. 2020. Bagging BERT models for robust aggression identification. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 55–61, Marseille, France. European Language Resources Association (ELRA).

Arjun Roy, Prashant Kapil, Kingshuk Basak, and Asif Ekbal. 2018. An ensemble approach for aggression identification in english and hindi text. In *TRAC-2018*, pages 66–73.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic Routing Between Capsules. *arXiv e-prints*, page arXiv:1710.09829.

Niloofar Safi Samghabadi, Deepthi Mave, Sudipta Kar, and Thamar Solorio. 2018. RiTUAL-UH at TRAC 2018 shared task: Aggression identification. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 12–18, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Niloofar Safi Samghabadi, Parth Patwa, Srinivas PYKL, Prerana Mukherjee, Amitava Das, and Thamar Solorio. 2020. Aggression and misogyny detection using BERT: A multi-task approach. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 126–131, Marseille, France. European Language Resources Association (ELRA).

Saurabh Srivastava, Prerna Khurana, and Vartika Tewari. 2018. Identifying aggression and toxicity in comments using capsule network. In *TRAC-2018*, pages 98–105.

Yequan Wang, Aixin Sun, Jialong Han, Ying Liu, and Xiaoyan Zhu. 2018. Sentiment analysis by capsules. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, pages 1165–1174, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Wei Zhao, Jianbo Ye, Min Yang, Zeyang Lei, Suofei Zhang, and Zhou Zhao. 2018. Investigating capsule networks with dynamic routing for text classification. *CoRR*, abs/1804.00538.

A New Approach to Claim Check-Worthiness Prediction and Claim Verification

Shukrity Si
Jalpaiguri Govt. Engg College
India
sukriti.si98@gmail.com

Anisha Datta
Jalpaiguri Govt. Engg College
India
dattaanishadatta@gmail.com

Sudip Kumar Naskar
Jadavpur University
India
sudip.naskar@cse.jdvu.ac.in

Abstract

The more we are advancing towards a modern world, the more it opens the path to falsification in every aspect of life. Even in case of knowing the surrounding, common people can not judge the actual scenario as the promises, comments and opinions of the influential people at power keep changing every day. Therefore computationally determining the truthfulness of such claims and comments has a very important societal impact. This paper describes a unique method to extract check-worthy claims from the 2016 US presidential debates and verify the truthfulness of the check-worthy claims. We classify the claims for check-worthiness with our modified Tf-Idf model which is used in background training on fact-checking news articles (NBC News and Washington Post). We check the truthfulness of the claims by using POS, sentiment score and cosine similarity features.

1 Introduction

Today we live in a world where falsehood seems to reflect everywhere be it in administration, sports, entertainment sector and even in the education field. Many popular and influential personalities seem to be vulnerable in keeping their words. The opinions and comments they make, their claims keep changing frequently. Therefore we can not blindly rely on present news. During the 2016 US presidential campaign, people came to realize how fake news could be spread in mainstream news channels and social media. (Alexandre Bovet, 2019) reported the influence of fake news on social media during the election. They showed that about 171 million tweets were made during the election among which 25% were fake or extremely biased. Many journalists started investigation into identifying the actual truth. However, it was a time consuming and tedious task to do the work manually. This problem gave rise to the concept of automatic fact and claim checking. Research has been going on since then to effectively tackle this problem which has proved to be a very challenging problem. Typically for the claim verification task, relevant evidence related to the claims is collected first and then the claim is compared with the evidence to know the actual fact. (Giovanni Luca Ciampaglia and Flammini., 2015) did this with the help of knowledge graph taken from Wikipedia.

We propose a suitable rule based approach with the help of feature engineering for the task. Our work consists of two tasks, first we extract the claims which are check-worthy and then we verify the truthfulness of these check-worthy claims. We carried out our experiments on the dataset of the Fact Checking Master (Preslav Nakov and Martino., 2018) shared task, organized in CLEF-2018, which deals with fact checking on the U.S. Presidential debate articles of 2016.

Extraction of check-worthy claims is carried out in two processes i.e. supervised and unsupervised approach. In unsupervised approach, the check-worthy claims are extracted with POS tags and K-Means Clustering algorithm. Dataset related to claims can be generated from any conversation with the help of this method. In supervised approach, we have collected some fact checking news articles (NBC News and Washington post) for background training and a modified Tf-Idf model is created to classify the claims whether check-worthy or not. Cosine similarity, Sentiment scores and POS tags are also used here. On comparing with the original labels, this model gives an accuracy of 98.6% when passed along with GBM.

Claim verification is performed by comparing the classified check-worthy claims with the fact checking news articles to verify their truthfulness. POS tagging, Cosine similarity are used to search for the explanations of the claims from the arti-

Proceedings of the 17th International Conference on Natural Language Processing, pages 155–160
Patna, India, December 18 - 21, 2020.

cles. With all these features together we make a hypothesis for the final classification.

These two tasks are consecutively done in a work (Pepa Gencheva and Koychev, 2017) previously. Else there are many works to influence on individual approaches of the model. The paper is divided into many sections, section 2 describes related works, dataset in section 3, features and proposed methodology in section 4 and 5 respectively, results in section 6 and conclusion is given in section 7.

2 Related Works

Fact-Checking has become a trending topic recently. Zhou and Zafarani (2018) provides a survey on fake news research and their study focuses on fake news from four perspective – the false knowledge it carries, its writing style, its propagation patterns, and the credibility of its creators and spreaders. Pepa Gencheva and Koychev (2017) extracted the check-worthy claims by comparing them with some popular news articles and verified the truthfulness of the claims using Support Vector Machines (SVM) and Feed-Forward Neural Networks (FNN).

Mihai Surdeanu and Manning (2010) used Conditional Random Field (CRF) for legal claim identification. Firstly they used Optical Character Recognition (OCR) to convert PDF documents into text and then they used four types of CRF architectures for the actual task. Datta and Si (2020) reported a work on fake news identification in which sentiment scores and Tf-Idf are used as features to build a Majority Voting model with four classifiers - Gradient Boosting, Random Forest, Extra Tree and XGBoost classifiers to identify fake news. Ghanem et al. (2019) showed that false news has various emotional patterns to mislead the readers and with the emotional sentiment features they propose an LSTM model for the classification task.

Suzuki and Takatsuka. (2016) proposed a keyword-extraction model for verifying patent claims. RoyBar-Haim and Slonim performed claim stance classification by automatic expansion of the initial sentiment lexicon and by using SVM with unigrams. Naeemul Hassan and Tremayne (2015) developed a system called ClaimBuster which monitors Twitter and retweets the check-worthy factual claims it finds and produces true - false verdicts for these types of factual claims. Moin Nadeem (2019) reported an end to end fact-checking system, FAKTA, using document retrieval from various media sources, evidence extraction and linguistic analysis. Dieu-Thu Le and Blessing (2016) used Convolutional Neural Networks for the task. Rob Ennals and Rosario. (2010) developed another fact-checking system, DisputeFinder, which works on already verified claims. Ayush Patwari and Bagchi. (2017) used LDA topic modeling, POS tuples and Bag-of-Words as features and SVM is used for clustering. Wang. (2017) and Nicole OBrien and Boix. (2018), proposed different models to classify factuality of claims aimed at only input claims and their metadata.

3 Dataset

We carried out our experiments on the dataset of the Fact Checking Master (Preslav Nakov and Martino., 2018) shared task, organized in CLEF-2018. The dataset contains stated claims of the U.S. Presidential debates (2016) with a total of 1,403 sentences in the first Presidential debate and 1,303 sentences in the second Presidential debate. There are four speakers - Holt (host of the first Presidential debate), Cooper (host of the second Presidential debate), Hillary Clinton and Donald Trump. All the claims made by Mr. Trump and Ms. Clinton in these debates can be sensed manually since all of the claims show some actions happened in the past, or any comments or actions from the opponent in the past. Observing the patterns, we analysed the prominent features and developed our models.

4 Proposed Methodology

4.1 Claim Check-Worthiness Prediction

In the dataset, there are many statements which should be prioritized to be fact-checked as claims. We emply both supervised and unsupervised approach for this task.

4.1.1 Supervised Approach

We used modified Tf-Idf model with Gradient Boosting classifier for the work. Term Frequency (tf) measures how frequently a term occurs in a document or text. Since every document is different in length, it is possible that a term can appear more frequently in the longer documents than in the shorter ones. Thus, term frequency is normalized by the document length, i.e., the total number of terms in the document.

Idf is generally defined by the logarithm of the ratio of total number of documents in the dataset and the number of documents with that term in

them. The idf calculation is modified in our model. We have taken the statements from the data as documents in Tf calculation. The modified Idf is defined as the logarithm of the ratio of total no. of documents (the explanations from the fact-checking articles) and number of documents with the term in them. In case of normal Tf-Idf count, the documents (or, sentences) used in both Tf and Idf belong to the same article. But in our case, Tf count takes the actual data but Idf count takes the fact-checking news articles for consideration. The reason behind this modification is to check word similarity between a claim and an explanation. The more the similarity is, the more an explanation is related to the claim. This process helps in background knowledge training.

Modified $Idf =$

$$log_e \frac{Total\ no.\ of\ documents\ in\ the articles}{No.\ of\ documents\ with\ the\ term}$$

But in case, if the term is not present in the Fact-Checking articles, the Idf count is taken as 0. Now the Tf and Idf is multiplied to calculate the modified Tf-Idf count(feature model).

$$Modified\ Tf-Idf = Tf * Modified\ Idf \quad (1)$$

Therefore we modify Tf-Idf in this way to build a feature matrix which can help us to establish a relation between the background articles and the statements.

If the calculated Tf-Idf is a non-zero number, then the term is present in the background article. This increases the chance of the statement with the term to be classified as a check-worthy claim as it is related to one of the explanations present in the articles. As this is a supervised approach, we use the labels of our data. In the data, if the statement is a check-worthy claim then it is assigned to 1 or otherwise 0. The Tf-Idf feature model is now fitted into Gradient Boosting classifier for final classification.

4.1.2 Unsupervised Approach

In this approach, our main goal is to extract the check-worthy claims from the debate dataset without any labels. This approach can be used in future to create new dataset related to claims from any debates, conversations or interviews. Here we study the data very carefully to understand the features of check-worthy claims which are described below.

Features - Now an example of a claim is -
"Ford is leaving, you see that, their small car division leaving, Thousands of jobs leaving Michigan, leaving Ohio."
In this case, 'leave' verb is in continuous tense, and there are proper nouns like 'Ford', 'Michigan', 'Ohio'. So the statements containing proper nouns and continuous tense have a great chance to be check-worthy claims.
"He approved NAFTA, which is the single worst trade deal ever approved in this country."
This is a claim made by Trump. The adjective 'bad' is in superlative form in this sentence, verb 'approve' is in past tense and a connective word 'which' is used here. Now if any person uses other person's statements in indirect speech with a connective word, then there is a strong possibility that the sentence is a check-worthy claim. Because the person may change other's statement in his own way, and the statement should be checked whether it was actually stated or not. Now if anyone says - "Paolo Coelho said that he was the best writer of the world." This sentence is a claim indeed. Paolo Coelho might say that he is one of the best writer of the world, but the person distorted his statement. So this type of claims are check-worthy.

Therefore with all these features, we have made separate matrices and merged them all together. This merged feature matrix is passed along with a unsupervised machine learning algorithm called 'K-means Clustering' to create two clusters of check-worthy claims and non-claims.

We have used these two approaches for the classification. Among them the supervised approach is more suitable for our work, it gives better results than the other. But the unsupervised approach can help us to generate claim dataset without any labels. But this model needs further modification.

4.2 Claim Verification

After extracting the check-worthy claims from the dataset, we are now left with 17 and 16 claims for the 1st and 2nd presidential debate respectively. These claims now need to be checked for truthfulness by comparing with the existing fact-checking articles. The second part of the dataset contains labels of the claims according to their truthfulness. There are 3 labels as True, False and Half-True. So, the next task is to make a suitable model to classify the check-worthy claims. Now this work is divided into two parts, first part is the extraction

of related explanations for all the claims and the second part is to verify whether the claims are true or not by comparing with the explanations. We have used NBC news and Washington post fact-checking articles to get the true explanations of the check-worthy claims. They are described below.

4.2.1 Explanation Extraction

The first goal of this task is to find the proper explanations of the claims. We have used Cosine Similarity algorithm with the help of POS tags.

- POS tags - We compare each check-worthy claim with all the explanations given in the NBC news and Washington post articles. The first step is to POS tag each sentence. Some of the tags are given more importance than the others. These are - nouns (proper nouns), normal pronouns and possessive pronouns, verbs (past, continuous and participle tenses), adjectives, adverbs and connectives ('that', 'which', etc.). These tags increase the chance of getting similar sentences. Therefore, all these POS tags are taken into consideration and a single feature matrix is formed by merging them all together.

- Cosine Similarity - We compute cosine similarity between combined POS tagged list of each of the check-worthy claims and the combined POS tagged list of each explanation of the fact-checking articles. The maximum output gives out the true explanation. Now the true explanations are placed beside the claims to check for the truthfulness.

4.2.2 Truthfulness Detection

The next task is to compare each check-worthy claim with its explanation and verify whether the claim is true or not. For this, we use sentiment scores and then build a new hypothesis which is used in the classification task.

Sentiment Score - We use VADER model (Hutto, 2014) to get the sentiment scores. It calculates the positive, negative, neutral and compound sentiment polarity for any sentence (in English language). With the help of this model, we have calculated the Sentiment Scores of each sentence, separated out the compound scores to be more precise and then compared the scores of each claim with its explanation. The compound score is calculated as in Equation 2, where, $a = (positive + negative + neutral)$ and α is a constant, say, $t = sentiment(claim) - sentiment(explanation)$.

$$Compound\ score = \frac{a}{\sqrt{a^2 + \alpha}} \tag{2}$$

Now, this needs to be standardised further to get the threshold value for every label (True, False and Half-True). We have separated the claims from the original dataset according to their labels. Then the calculated t is placed accordingly, their means and medians are calculated with respect to the labels.

Observing the results, we have chosen threshold values for each label. The values are chosen as given below.

If $t \in [0.40, \infty)$, then the claim is True.

If $t \in [0.20, 0.40)$, then the claim is Half-True.

If $t \in [-\infty, 0.20)$, then the claim is False.

We determine the threshold by calculating mean and median of the variables, but if we look into the scores, we can understand that the threshold should work in right way. If any claim is false, generally its explanation will carry negative words like 'he didn't', 'it's a lie' etc. As we subtract the sentiments, for false claims, it becomes negative. And for true claims, the value is in positive range and for half true it lies in between them. This is the hypothesis we propose for claim verification.

This gives new labels for the claims. We have compared them with the existing labels and got reasonable scores for each label.

5 Results and Discussion

We calculate the values of accuracy, precision, recall and f-score for each model proposed in the paper. Confusion matrices are also shown for result visualization for both the debate articles. All the results are analysed below.

5.1 Claim Check-worthiness

We work on two dataset, 1st and 2nd US presidential candidate debate articles. We have used a modified version of Tf-Idf (a new method we have proposed here) model to extract check-worthy claims. We have tested the model with the original labels of the dataset with the help of GBM classifier. The results for both datasets are given in the table 1 and table 2.

Figure 1: Confusion Matrix of 1st debate article results

Figure 2: Confusion Matrix of 2nd debate article results

Table 1: Results on 1st-Presidential Debate-

Model	Accuracy	Precision	Recall	F-Score
GBM	98.65%	98.66%	98.64%	98.42%

Table 2: Results on 2nd-Presidential Debate-

Model	Accuracy	Precision	Recall	F-Score
GBM	99%	99.01%	99%	98.86%

Now we can see that the results of both debates are very good. The confusion matrices for the models on 1st and 2nd dataset are shown in figure 1 and 2 respectively.

Now there arise some cases of wrong predictions. A few check-worthy claims are present which cannot be extracted by our model but most of the predictions are correct. But there are no such extracted claims which are not originally check-worthy. Here we can conclude that our model is giving a very good performance for the check-worthiness problem. Although we will work on it for improvement.

5.2 Claim Verification

We have used some features like- POS tagging, Cosine Similarity and Sentiment Score on the extracted check-worthy claims to extract the true explanations from the fact-checking articles and verify the truthfulness of the claims. This approach on both the debate articles has brought good results for the labels - True and False. But for label Half-True, for some ambiguity, the result is not so good like the others. For this reason, the overall result has decreased. The individual results for each label and on a whole are shown below in the table 3, 4, 5, 6.

Results on 1st-Presidential Debate

Table 3:

Accuracy	Precision	Recall	F-score
64.705%	74.50%	64.70%	62.64%

Table 4:

	True	False	Half-True
Accuracy	75%	88%	33.3%

Results on 2nd-Presidential Debate

Table 5:

Accuracy	Precision	Recall	F-score
62.5%	67.05%	62.5%	61.25%

Table 6:

	True	False	Half-True
Accuracy	50%	77.7%	33.3%

All the predictions of our model is discussed and we realise that the models are facing some difficulties to classify the class half true. But if the half-true label is considered as false then the result is fine. But we are working on the improvement of this model for the particular class half true.

6 Conclusion and Future Work

The method which we use gives a very good result in check-worthiness classification, and the approach we propose here is a modified version of Tf-Idf for using background fact checking article. This can help us to do further research in background training. Other approach we have used is unsupervised approach, where we studied every feature very carefully and built a model. Though this model needs further modification but it can be used to generate new data on claims from any debate, conversation or interview. So it can contribute to the field of dataset creation. Now for Claim truthfulness, we first extracted the proper explanations from the articles, then applied our own threshold for classification. This approach works good for the class true and false, but for half true, it needs tuning and modification.

Further we can apply our modified Tf-Idf model in other researchers' works and check the efficiency. The model and hypothesis for truthfulness verification needs further improvement. We will try to add deep learning methods also. It can be concluded that our proposed model is very versatile and can be used in other fields as well.

References

Hernan A. Makse. Alexandre Bovet. 2019. Influence of fake news in twitter during the 2016 us presidential election. ArXiv:1803.08491v2 [cs.SI].

Dan Goldwasser Ayush Patwari and Saurabh Bagchi. 2017. Tathya:amulti-classifier system for detecting check-worthy statements in political debates. In *In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. Singapore, CIKM '17, pages 2259 to 2262.*

Anisha Datta and Shukrity Si. 2020. A supervised machine learning approach to fake news identification. In *Intelligent Data Communication Technologies and Internet of Things*, pages 197–204, Cham. Springer International Publishing.

Ngoc Thang Vu Dieu-Thu Le and Andre Blessing. 2016. Towards a text analysis system for political debates. on. LaTeCH.

Bilal Ghanem, Paolo Rosso, and Francisco M. Rangel Pardo. 2019. An emotional analysis of false information in social media and news articles. *ArXiv*, abs/1908.09951.

Luis M. Rocha-Johan Bollen Filippo Menczer Giovanni Luca Ciampaglia, Prashant Shiralkar and Alessandro Flammini. 2015. Computational fact checking from knowledge networks. PLOS ONE 10(6):1 to 13.

E.E. Hutto, C.J. Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. eighth international conference on weblogs and social media. *(ICWSM-14). Ann Arbor, MI, June 2014.*

Ramesh Nallapati Mihai Surdeanu and Christopher Manning. 2010. Legal claim identification: Information extraction with hierarchically labeled data. LREC.

Brian Xu Mitra Mohtarami-James Glass. Moin Nadeem, Wei Fang. 2019. Fakta: An automatic end-to-end fact checking system. In *Proceedings of NAACL-HLT 2019: Demonstrations, pages 78 to 83 Minneapolis, Minnesota, June 2 to June 7, 2019. Association for Computational Linguistics.*

Chengkai Li Naeemul Hassan and Mark Tremayne. 2015. Detecting check-worthy factual claims in presidential debates. In *In Proceedings of the 24th ACM International Conference on Information and Knowledge Management. CIKM '15, pages 1835–1838.*

Georgios Evan-gelopoulos Nicole OBrien, Sophia Latessa and Xavier Boix. 2018. The language of fake news: Opening the black-box of deep learning based detectors. In *InProceedings of the Thirty-second Annual Conference on Neural Information Process-ing Systems (NeurIPS)–AI for Social Good.*

Lluıs M'arquez-Alberto Barron-Cede˜no Pepa Gencheva, Preslav Nakov and Ivan Koychev. 2017. A context-aware approach for detecting worth-checking claims in political debates. RANLP. Varna, Bulgaria, pages 267–276.

Tamer Elsayed Reem Suwaileh Lluıs M'arquez Wajdi Zaghouani Pepa Atanasova Spas Kyuchukov Preslav Nakov, Alberto Barron-Cede˜no and Giovanni Da San Martino. 2018. Overview of the clef 2018 checkthat!lab on automatic identfication and verification of political claims. In *In Proceedings of CLEF. Avignon, France, pages 372 to 387.*

John Mark Agosta Rob Ennals, Dan Byler and Barbara Rosario. 2010. What is disputed on the web?. In *In Proceedings of the 4th workshop on Information credibility. ACM, New York, NY, USA, WICOW '10, pages 67 to 74.*

Charles Jochim RoyBar-Haim, Lilach Edelstein and Noam Slonim. Improving claim stance classification with lexical knowledge expansion and context utilization. In *Proceedings of the 4th Workshop on Argument Mining, pages 32–38 Copenhagen, Denmark, September 8, 2017. c 2017 Association for Computational Linguistics*.

Shoko Suzuki and Hiromichi Takatsuka. 2016. Extraction of keywords of novelties from patent claims. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1192–1200, Osaka, Japan, December 11 to 17 2016.*

William Yang Wang. 2017. iar, liar pants on fire":a new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume2: Short Papers), pages 422 to 426. Association for Computational Linguistics.*

Xinyi Zhou and Reza Zafarani. 2018. Fakenews: A survey of research, detection methods, and opportunities. ArXiv:1812.00315v1 [cs.CL].

Improving Passage Re-Ranking with Word N-Gram Aware Coattention Encoder

Chaitanya Sai Alaparthi and **Manish Shrivastava**
Language Technologies Research Centre (LTRC),
Kohli Centre on Intelligent Systems(KCIS),
International Institute of Information Technology, Hyderabad, India
chaitanyasai.alaparthi@research.iiit.ac.in
m.shrivastava@iiit.ac.in

Abstract

In text matching applications, coattentions have proved to be highly effective attention mechanisms. Coattention enables the learning to attend based on computing word level affinity scores between two texts. In this paper, we propose two improvements to coattention mechanism in the context of passage ranking (re-ranking). First, we extend the coattention mechanism by applying it across all word n-grams of query and passage. We show that these word n-gram coattentions can capture local context in query and passage to better judge the relevance between them. Second, we further improve the model performance by proposing a query based attention pooling on passage encodings. We evaluate these two methods on MSMARCO passage re-ranking task. The experiment results shows that these two methods resulted in a relative increase of 8.04% in Mean Reciprocal Rank @10 (MRR@10) compared to the naive coattention mechanism. At the time of writing this paper, our methods are the best non transformer model on MS MARCO passage re-ranking task and are competitive to BERT base while only having less than 10% of the parameters.

1 Introduction

Passage ranking (or re-ranking) is a key information retrieval (IR) task in which a model has to rank (or re-rank) set of passages based on how relevant they are to a given query. It is an integral part of conversational search systems, automated question answering systems (QA), etc., The typical answer extraction process in these systems consists of two main phases. The first phase is ranking passages from the collection that most likely contain the answers. The second phase is extracting answers from these passages. The performance of first phase significantly impact the performance of extracting answers and the performance of the overall system. Thus it is important for a QA system to effectively rank passages.

Attention mechanisms have shown tremendous improvements in the deep learning based NLP models (Bahdanau et al., 2014; Wang et al., 2016; Yang et al., 2016; Lu et al., 2016; Vaswani et al., 2017). Attention allows the model to dynamically focus only on certain parts of the input that helps in performing the task at hand effectively. Coattentions (Xiong et al., 2016) are class of attention mechanisms which can be applied on text matching problems. They proved to be highly effective as they enables the learning to attend based on computing word level affinity scores between two texts thus helping in effectively deciding the relevance between them.

This paper builds on previous work on coattention mechanism (Alaparthi, 2019) (we call it as naive coattention encoder) to tackle the problem of passage re-ranking. We recall that the coattention encoder attends across query and passage by computing the *word-level* affinity scores. Similar to (Hui et al., 2018), we argue that attending at *word-level* limits the ability to capture local context in the query-passage interactions. As an example (which we later explain in section 5.3), for a query: *what is January birthstone color*, the naive coattention encoder can relate the passages describing passages such as *November birthstone color*, *April birthstone color*, etc. This is likely because of common matching terms *birthstone* and *color* and semantically similar words *January*, *November*, *April*, etc. We demonstrate that extending the coattentions to words and n-grams can improve the matching signals, which will contribute to final relevance scores.

In the naive coattention encoder, *max-pooling* was applied on the coattention encodings to obtain the co-dependent representation of the passage,

Proceedings of the 17th International Conference on Natural Language Processing, pages 161–169
Patna, India, December 18 - 21, 2020.

which forms the base in deciding the relevance. We argue that using max-pooling limits the ability to compose complex co-dependent representations. Intuitively, we can leverage query to supervise the co-dependent representation from coattention encodings of the passage. With this intuition, we propose a simple query-based attention pooling. We show that query-based attention pooling can pick the appropriate clauses which are distributed across the coattention encodings. This allows the model to only focus on relevant parts of passage coattention encodings, which are appropriate in judging the relevance. Additionally, the final passage representation is supervised by the query, which helps the model to better judge the relevance.

To solve these challenges, we first extend the coattention encoder to words and phrases by applying it across all word n-grams of query and passage. For this purpose, similar to C-LSTM (Zhou et al., 2015) and Conv-KNRM (Dai et al., 2018), we generate the word n-gram representations from the word embeddings of query and passage using the convolutional layers of different heights and multiple filters. We then encode these n-gram sequences using a BiLSTM to capture the long term dependencies into n-gram encodings. A coattention encoder is then applied on these n-gram encodings of query and passage to get the coattention encoding of the passage. The coattention encoder first generates the co-dependent representations of query and passage by attending across all word n-grams from query and passage. These co-dependent representations of passage are then fused with the n-gram encodings of the passage using a BiLSTM to get the coattention encoding of the passage. We show that this coattention encoding can better capture the local context between query and passage, thus improving the overall judging power of the model.

To get the final representation of the passage, Alaparthi (2019) applied max-pooling over time on the coattention encoding of the passage (note that coattention encoding is the outputs of BiLSTM). This final representation forms the base in deciding the relevance between query and passage. In this paper, we apply a query based attention pooling on the coattention encoding instead of max-pooling to pick the appropriate clauses which are distributed across the coattention encoding. We argue that, query based attention pooling allows the model to only focus on relevant parts of passage coattention encoding which are appropriate in judging the relevance. Additionally, the final passage representation is supervised by query, which helps the model to better judge the relevance.

We experimented our methods on MS MARCO passage re-ranking task [1] (Bajaj et al., 2016). Making the coattention encoder n-gram aware (uni,bigrams) has increased the Mean Reciprocal Rank @10 (MRR@10) from 28.6 to 29.9 (+4.5% relative increase) when compared to the naive coattention encoder. Replacing the max pooling layer with the query based attention pooling has further improved the MRR@10 to 30.9 (+8.08% overall relative increase), resulting in the best non transformer based model. We show that our methods are competitive to BERT base despite having very less number of parameters. Also, our methods can be easily trained and requires much lesser computational resources.

To summarize, the key contributions of this work are as follows: First, we extend the naive coattention encoder to words and phrases making the coattentions to capture local context. We call it as n-gram coattention encoder. Second, we further extend the n-gram coattention encoder with query based attention pooling to pick the appropriate clauses which are distributed across the coattention encoding of the passage. We call it as n-gram coattention encoder with attention pooling. We show that this can further improve the model in deciding the relevance. Third, we apply our methods on MS MARCO passage re-ranking task and show that our methods have outperformed all the baselines including the previous best non BERT model and are competitive to BERT base. Last, we use examples to compare and discuss our methods with naive coattention encoder.

In section 2, we discuss related work. Then in section 3, we describe our two methods of improving naive coattention encoder. In section 4, we describe the dataset, baselines and the settings we used in all our experiments. Next, we analyze and discuss the results in section 5. Finally, we conclude our work with future plans in section 6.

2 Related Work

Deep learning methods have been successfully applied to a variety of language and information retrieval tasks. By exploiting deep architectures, deep learning techniques are able to discover from training data the hidden structures and features at dif-

[1] https://github.com/microsoft/MSMARCO-Passage-Ranking

ferent levels of abstractions useful for the tasks. Therefore a new direction of Neural IR is proposed to resort to deep learning for tackling the feature engineering problem of learning to rank, by directly using only automatically learned features from raw text of query and passage.

The first successful model of this type is Deep Structured Semantic Model (DSSM) (Huang et al., 2013) introduced in 2013, which is a neural ranking model that directly tackles the adhoc retrieval task. In the same year Lu and Li proposed DeepMatch (Lu and Li, 2013) which is a deep matching method applied to the community question answering and micro-blog matching tasks. Later from 2014 and 2015, there is a rapid increase in neural ranking models, such as new variants of DSSM (Shen et al., 2014), ARC I and ARC II (Hu et al., 2014), MatchPyramid (Pang et al., 2016), etc.,

With the introduction of large scale datasets such as MS MARCO (Bajaj et al., 2016), we have seen a tremendous improvements in neural ranking models. Well known architectures include DUET (Mitra et al., 2017), DUET V2 (Mitra and Craswell, 2019), KNRM (Xiong et al., 2017), Conv-KNRM (Dai et al., 2018), Coattention encoder (Alaparthi, 2019) including the transformer based architectures such as BERT (Nogueira and Cho, 2019), DuoBERT (Nogueira et al., 2019), RepBERT (Zhan et al., 2020).

3 Methodology

In this section, we first describe with notations. Next in section 3.2, we briefly describe the naive coattention encoder first proposed in (Xiong et al., 2016). In section 3.3 and 3.4, we describe our two methods to improve the naive coattention encoder.

Notations Let $Q^{emb} = (x_1^Q, \ldots, x_n^Q) \in \mathbb{R}^{n \times L}$ be the embeddings of words in query of length n, where each word embedding is of dimension L. Similarly, $P^{emb} = (x_1^P, \ldots, x_m^P) \in \mathbb{R}^{m \times L}$ denote the same for words in passage of length m.

3.1 Naive Coattention Encoder

Coattention encoder can be applied on Q^{emb} and P^{emb} to get the coattention encoding of the passage. We first start with encoding the Q^{emb} and P^{emb} using the same BiLSTM (Mueller and Thyagarajan, 2016) to share the representational power:

$$q_t = BiLSTM(q_{t-1}, x_t^Q) \quad (1)$$

and

$$p_t = BiLSTM(p_{t-1}, x_t^P) \quad (2)$$

Similar to (Merity et al., 2016; Xiong et al., 2016), we also add sentinel vectors q_ϕ, p_ϕ to allow the query to not attend to any particular word in the passage. So $Q = (q_1, \ldots, q_n, q_\phi) \in \mathbb{R}^{(n+1) \times L}$ and similarly $P = (p_1, \ldots, p_m, p_\phi) \in \mathbb{R}^{(m+1) \times L}$.

Next, we compute the affinity scores between all pairs of query and passage words: $L = P^T Q$. We call L as affinity matrix. The affinity matrix is normalized row wise to get the attention weights A^Q across the passage for each word in query. Similarly, normalized column wise to get the attention weights A^P across the query for each word in the passage:

$$A^Q = softmax(L) \quad (3)$$

and

$$A^P = softmax(L^T) \quad (4)$$

Next, we compute the attention contexts, of the passage in light of each word in the query:

$$C^Q = PA^Q \quad (5)$$

Additionally, we compute the summaries $C^Q A^P$ of the previous attention contexts in light of each word in the passage. We also define C^P, a co-dependent representation of the query and passage, as the coattention context:

$$C^P = [Q; C^Q]A^P \quad (6)$$

Here $[x; y]$ is concatenation of vectors x and y horizontally. The last step is the fusion of temporal information to the coattention context via a bidirectional LSTM:

$$u_i = BiLSTM_{fusion}(u_{i-1}, u_{i+1}, [p_i; c_i^P]) \quad (7)$$

We define $U = [u_1, \ldots, u_m]$, the outputs of $BiLSTM_{fusion}$ concatenated vertically, as the coattention encoding of the passage. Here $U \in \mathbb{R}^{m \times L'}$ and L' is the dimension of the hidden state in $BiLSTM_{fusion}$.

For the rest of this paper, we treat the coattention encoder as a module, defined as:

$$U = CoAttentionEncoder(Q^{emb}, P^{emb}) \quad (8)$$

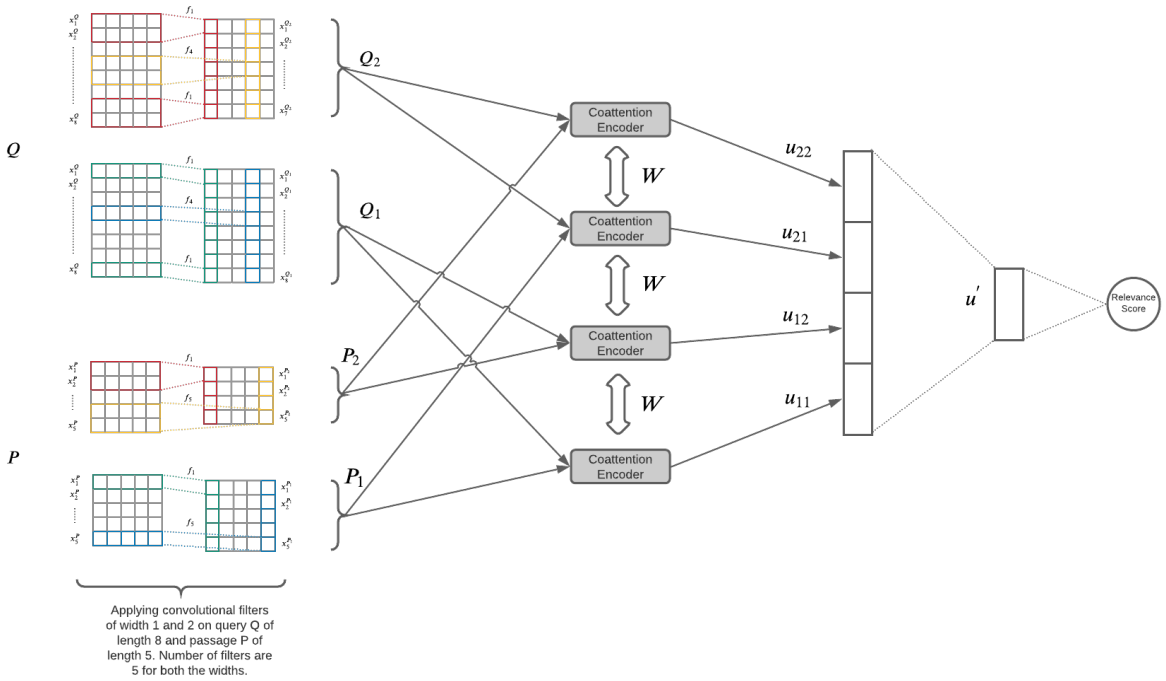


Figure 1: Architecture of ngram aware coattention encoder

3.2 Word N-Gram Coattention Encoder

In this section, we extend the naive coattention encoder by applying it across all word n-grams from query and passage as shown in Figure 1. To compute the word h-gram representations of query Q^{emb}, where each h-gram representation is of dimension F, similar to (Zhou et al., 2015; Dai et al., 2018), we apply F convolution filters of height h and width L. Note that L is the dimension of the word embeddings. For each window of h words, a single filter $filter_f$ performs a weighted sum of all word embeddings $x^Q_{t:t+h}$ parameterized by it's weights $w_f \in \mathbb{R}^{hL}$ and bias $b_f \in \mathbb{R}$:

$$v^h_f = w_f \cdot x^Q_{t:t+h} + b_f, v_f \in \mathbb{R} \quad (9)$$

Using F filters, we get F scores $v^h_1, .., v^h_F$, each describing $x^Q_{t:t+h}$ in a different perspective. These v^h_f from F filters are concatenated into a single vector and $tanh$ activation is then applied to get the F-dimensional embedding:

$$x^{Q_h}_t = tanh([v^h_1; \ldots; v^h_F]) \in \mathbb{R}^F, t = 1..n-h+1 \quad (10)$$

We define h-gram sequence of the query as

$$Q_h = [x^{Q_h}_1, ..., x^{Q_h}_{n-h+1}] \quad (11)$$

Note that padding is not applied to the sequence. Similarly, we apply the same convolution filters to get the h-gram representations of passage P^{emb}:

$$P_h = [x^{P_h}_1, \ldots, x^{P_h}_{m-h+1}] \quad (12)$$

Here $Q_h \in \mathbb{R}^{(n-h+1)\times F}$ and $P_h \in \mathbb{R}^{(m-h+1)\times F}$. Using these convolutional layers of different heights, we get different n-gram sequences.

Coattention encoder is applied on all Q_i and P_j $\forall i, j \in [1..H]$, H is a parameter, which denotes the span of the n-gram. In this paper, we have only experimented with uni,bi-grams i.e., $H = 2$. Note that, $H = 1$ reduces to naive coattention encoder i.e., unigrams/words. Coattention encoder applied on Q^i and P^j generates a coattention encoding of the passage denoted by:

$$U_{ij} = CoAttentionEncoder(Q_i, P_j) \quad \forall i, j \in [1..H] \quad (13)$$

Here $U_{ij} \in \mathbb{R}^{(m-j+1)\times L'}$ and to recall,

$$U_{ij} = [u_{ij_1}, u_{ij_2}, \ldots, u_{ij_{m-j+1}}] \quad (14)$$

where u_{ij_t} is the output from the $BiLSTM_{fusion}$ at time step t. We call U_{ij} as the coattention encoding of the passage P_j with respect to the query Q_i.

To get the relevance score, similar to (Alaparthi, 2019), a max-pooling layer over time can be applied on U_{ij} to get the single representation (single

thought vector):

$$u_{ij} = max(\{u_{ij_t}\}_{t=1..m-j+1}) \in \mathbb{R}^{L^{'}} \quad (15)$$

We concatenate these representations $u_{ij} \forall i, j \in [1..H]$ horizontally to get a single vector $u^{'}$ a n-gram aware coattention representation of passage.

$$u^{'} = [u_{11}; u_{12}; \ldots; u_{H-1H}; u_{HH}] \in \mathbb{R}^{H^2 L^{'}} \quad (16)$$

The $u^{'}$ is then passed to a linear layer parameterized by weights $W_s \in \mathbb{R}^{H^2 L^{'}}$ to get the relevance score:

$$score_{P|Q} = W_s^T u^{'} \quad (17)$$

3.3 Coattention Encoder with Attention Pooling

In the naive coattention encoder and in the previous section, max-pooling over time is applied on the coattention encoding to get the single representation capturing entire sense of the passage. In this section, we propose a simple attention pooling to select key parts from the coattention encoding U_{ij} of the passage using the query: $q^{'} = q_{i_{n^{'}}}$, $n^{'} = n - i + 1$. We also add sentinel vector $d_{\phi}^{'}$ (Merity et al., 2016) to U_{ij} to allow the query to not attend to any particular clause in the passage:

$$u_{ij} = \sum_{t=1}^{t=m-j+1} \alpha_t u_{ij_t} \quad (18)$$

Where,

$$\alpha_t = \frac{exp(u_{ij_t}^T q^{'})}{\sum_{k=1}^{k=m-j+1} exp(u_{ij_k}^T q^{'})} \quad (19)$$

Similar to previous section, $u_{ij} \forall i, j \in [1..H]$ can then be concatenated horizontally into a single vector $u^{'}$ and this $u^{'}$ can be used to get the relevance score.

4 Experimental Setup

This section describes our datasets, how training and testing were performed and our implementation details.

4.1 Dataset and Learning rule

We perform all our experiments on Microsoft MAchine Reading COmprehension (MS MARCO) passage re-ranking task. The whole corpus consists of 8.8M passages extracted from 3.6M web documents corresponding to 500K anonymized user queries sampled from Bing's search query logs.

For the ease of training, MS MARCO team has released a pre-processed training set *triples.train.small.tsv*[1], which contain the triples $\langle Q, P^+, P^- \rangle$, where Q is the query, P^+ and P^- are passages, P^+ being more relevant. We train all our models on *triples.train.small.tsv*[2]. We use the Cross Entropy loss employed by a softmax function on relevance scores $score_{P^+|Q}$ and $score_{P^-|Q}$ to learn the parameters Θ of the models. Some subset of query, passages are randomly chosen from *top1000.dev.tsv*[2] to tune the models. Finally, we predict the ranks on *top1000.eval.tsv*[2].

$$L(\Theta) = -\sum_{\langle Q, P^+, P^- \rangle \in \mathbf{S}} logP(P^+|\langle Q, P^+, P^- \rangle)) \quad (20)$$

where,

$$P(P^+|\langle Q, P^+, P^- \rangle)) = \frac{exp(score_{P^+|Q})}{exp(score_{P^+|Q}) + exp(score_{P^-|Q})} \quad (21)$$

4.2 Hyperparameters

In all our experiments, we use FastText (Bojanowski et al., 2017) word embeddings of dimension 300. These FastText embeddings are trained on all queries and passages from the training set, we freeze these embeddings during the training. All the other parameters of the model are initialized using an uniform distribution $U(-0.01, 0.01)$. The number of filters in convolution layers is set to 300. We only experiment with uni and bi-grams i.e, $H = 2$. We use the BiLSTMs with 2 layers and hidden sizes of 512 with dropout of 0.2 (Srivastava et al., 2014) between the layers. ADAM optimizer (Kingma and Ba, 2014) with initial learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$ is used. We truncate the query and passage lengths to 30 and 150 words, train our network until convergence with batch size of 128. On a single 1080 Ti machine, training takes around 8 hours to converge. We evaluate our model on dev set every 500 steps and decay the learning rate by a factor of 0.5 every 5,000 steps.

[2] https://github.com/microsoft/MSMARCO-Passage-Ranking#data-information-and-formating

Method	MRR@10 Dev	MRR@10 Eval	Parameters
KNRM (Xiong et al., 2017)	21.8	19.8	-
Duet V2 (Mitra and Craswell, 2019)*	24.3	24.5	-
Conv-KNRM (Dai et al., 2018)	24.7	24.7	-
FastText + Conv-KNRM (Hofstätter et al., 2019)	29.0	27.1	-
IRNet **	27.8	28.1	-
Naive coattention encoder (Alaparthi, 2019)	28.8	28.6	6.9M§
N-gram coattention encoder **(Ours)**	31.0	29.9 **(+4.54%)**	9.6M §
+ attention pooling **(Ours)**	31.9	30.9 **(+8.08%)**	9.6M §
BERT Base	34.4	33.5	109M
BERT Large (Nogueira and Cho, 2019)	36.5	35.8	340M

Table 1: Comparison of the different methods. The variants of the coattention encoder benefits significantly from the modifications described in this paper. * Official Baseline; ** Unpublished work; § These do not include the parameters from word embeddings as we directly use the pre-trained FastText embeddings and do not update them during the training.

5 Results and Discussion

In this section, we present the evaluation results of our models and compare our models with various baselines.

5.1 Comparision with Baselines

Table 1 lists the MRR@10 scores on Dev and Eval sets. We compare the naive coattention encoder and two proposed methods with the baselines including BERT. From the table, we get the following observations: (1) Firstly, the naive coattention encoder has performed better than the existing best non BERT based models: Conv-KNRM and IRNet (2) Applying the coattention encoder on uni-grams and bi-grams resulted in an increase in MRR@10 on eval from 28.6 to 29.9 (relative increase of 4.5%). This indicates that the model can capture the more robust interactions between query and passage. (3) Using the query based attention pooling instead of max-pooling over time further increased the score by 3.3% indicating that the model can now focus on the appropriate clauses in the passage leading to better passage representation and appropriate relevance score. (4) We can also observe that despite having less number of parameters compared to BERT, (9.6M vs. 109M), our models are competitive to BERT (30.9 vs. 33.5).

5.2 Analysis with respect to query type

Table 2 lists the MRR@10 scores of naive coattention encoder (represented by **A**), n-gram coattention encoder (represented by **B**) and n-gram coattention encoder with attention pooling (represented by **C**) which were evaluated using Dev set.

Type	# Queries	A	B	C
what	2751	28.07	29.25	**30.85**
others	2274	31.21	32.17	**33.86**
how	837	23.28	23.8	**25.73**
where	283	37.6	37.69	**39.32**
who	278	28.9	29.91	**33.66**
when	189	23.42	**26.38**	23.92
define	173	27.84	**29.21**	28.31
which	120	21.03	22.35	**23.53**
why	75	19.55	**23.31**	22.92

Table 2: Comparision of naive coattention encoder (**A**) with the two variants described in this paper with respect to query type. In the table, **B** represents the n-gram coattention encoder and **C** represents the n-gram coattention encoder with attention pooling.

From the table, we can observe that, both the n-gram coattention encoders consistently outperformed the naive coattention encoder. It is interesting to see that plain n-gram coattention encoder (with out attention pooling, represented by column **B**) outperformed the n-gram coattention encoder with attention pooling (column **C**) in case of *when*, *define*, *why* type queries.

5.3 Qualitative Analysis

Table 3 lists the best passages ranked by the naive coattention encoder and our methods described in this paper for the various queries. In this section, we qualitatively analyze the performances of the coattention encoders.

Query	Method	Best Ranked passage
who is tom cavanagh?	Naive coattention encoder	For other people named Tom Corbett, see Tom Corbett (disambiguation). Thomas Wingett Tom Corbett, Jr. (born June 17, 1949) is an American politician and attorney who served as the 46th Governor of Pennsylvania from January 18, 2011 to January 20, 2015. He is a member of the Republican Party.
	N-gram coattention encoder	Tom Cavanagh on Why Grant Gustin Deserves to Be THE FLASH in the Movies, Too. Share: Tom Cavanagh is a national treasure. No, not just because he is the Tom in our beloved Mike and Tom Eat Snacks podcast, or because of his chilling performance as Dr. Harrison Wells on The CW's The Flash. But rather because the Canadian actor is unafraid to speak his mind — which often happens to coincide with exactly what we were thinking, too.
	N-gram coattention encoder with query attention	Grodd (via Harrison Wells) Thomas Patrick Tom Cavanagh (born October 26, 1963) is a Canadian actor. He portrays the various iterations of Harrison Wells on The Flash.
what is January birthstone color	Naive coattention encoder	November Birthstone Color. The November birthstone color is usually light to dark yellow, however, topaz, the official November birthstone comes in a range of great colors such as several shades of yellow, pale green, blue, red, pink, black, and brown. Pure topaz is actually a colorless stone. The red and pink topaz gets their color from chromium.
	N-gram coattention encoder	Birthstone color list. January Birthstone Color. The birthstone for the month of January is the garnet, which means that red is the commonly accepted January birthstone color. It signifies trust and friendship, which makes it a good gift for a friend. The word garnet comes from the Latin word granatum, which means pomegranate.
	N-gram coattention encoder with query attention	Birthstone color list. January Birthstone Color. The birthstone for the month of January is the garnet, which means that red is the commonly accepted January birthstone color. It signifies trust and friendship, which makes it a good gift for a friend. The word garnet comes from the Latin word granatum, which means pomegranate.
why was napalm used in the vietnam war	Naive coattention encoder	Napalm. Napalm is jellied gasoline. Its name is an acronym of naphthenic and palmitic acids, which are used in its manufacture. While used in World War II and the Korean War, napalm became notorious in Vietnam where it was used in three capacities.Possibly its most visual use was being dropped from aircraft in large canisters which tumbled sluggishly to earth.apalm is jellied gasoline. Its name is an acronym of naphthenic and palmitic acids, which are used in its manufacture.
	N-gram coattention encoder	Napalm. U.S. troops used a substance known as napalm from about 1965 to 1972 in the Vietnam War; napalm is a mixture of plastic polystyrene, hydrocarbon benzene, and gasoline. This mixture creates a jelly-like substance that, when ignited, sticks to practically anything and burns up to ten minutes.
	N-gram coattention encoder with query attention	The US first used napalm during World War II in both the European and Pacific theaters, and also deployed it during the Korean War. However, those instances are dwarfed by American use of napalm in the Vietnam War, where the US dropped almost 400,000 tons of napalm bombs in the decade between 1963 and 1973. Of the Vietnamese people who were on the receiving end, 60% suffered fifth degree burns, meaning that the burn went down to the bone.
what energy source is earth using primarily for its internal heat	Naive coattention encoder	5. According to the lecture, what energy source is Earth using primarily for its internal processes? a. [Interior heat] b. [Geothermal energy] c. [Solar energy] d. [Radioactive Decay] e. [Magma] 6. According to the lecture, what energy source is Earth using primarily for its external/surficial processes? a. [Interior heat] b. [Geothermal energy] c. [Solar energy] d.
	N-gram coattention encoder	5. According to the lecture, what energy source is Earth using primarily for its internal processes? a. [Interior heat] b. [Geothermal energy] c. [Solar energy] d. [Radioactive Decay] e. [Magma] 6. According to the lecture, what energy source is Earth using primarily for its external/surficial processes? a. [Interior heat] b. [Geothermal energy] c. [Solar energy] d.
	N-gram coattention encoder with query attention	5. According to the lecture, what energy source is Earth using primarily for its internal processes? a. [Interior heat] b. [Geothermal energy] c. [Solar energy] d. [Radioactive Decay] e. [Magma] 6. According to the lecture, what energy source is Earth using primarily for its external/surficial processes? a. [Interior heat] b. [Geothermal energy] c. [Solar energy] d.

Table 3: Best ranked passages by naive coattention encoder and it's 2 variants

Considering the query *Who is tom cavanagh?*, we can notice that naive coattention encoder although ranked the passage which semantically answers the query, it utterly fails as Tom Corbett and Tom Cavanagh are completely different persons. Although n-gram aware coattention encoder was able to mark the passage containing Tom Cavanagh as relevant, but it could not correctly capture the required sense from the passage. Finally, adding the query attention to the n-gram coattention encoder improved the ranking performance as we can see that the model was now able to correctly rank the passage.

Similarly, in case of query *what is January birthstone color*, naive coattention encoder has marked the passage relevant which corresponds to November birthstone color. However, both the n-gram coattention encoders have marked the correct passage as relevant, which is related to *January birthstone*. These two examples suggests that n-gram coattention encoders are able to correctly capture the local context and can capture the robust interactions between query and passages, thus improving the overall model performance.

In case of third query *why was napalm used in the vietnam war*, the naive coattention encoder predicted the passage containing the terms napalm, vietnam war. But the passage does not answer the query. The n-gram coattention with attention pooling marks the passage as relevant which describes about the effects napalm has created in vietnam war but the passage does not correctly answer the reason for using napalm in vietnam war. Interestingly, the n-gram coattention with out attention pooling predicted the correct relevant passage.

Lastly, for the query *what energy source is earth using primarily for its internal heat*, all the coattention encoders predicted a passage which is not

relevant. One interesting observation is that, the query is part of the passage itself. This shows that the coattention mechanism has trouble discriminating the passage which is semantically similar to the query but does not have an answer in it.

6 Conclusion and Future Work

In this paper, we proposed two simple extensions to naive coattention encoder, namely, n-gram coattention encoder which attends the words and word n-grams to better capture the interactions between query and passage. Later, we proposed simple attention pooling to pick the appropriate clauses which are distributed across the coattention encoding of the passage. Our experiments on MS MARCO passage re-ranking task shows that our models outperformed all the baselines including the naive coattention encoder. We also compare our methods with BERT and show that our methods are competitive to BERT base despite having very less number of parameters, thus our models are very easy to train and are computationally efficient.

We have also compared the performance of coattention encoders with respect to the query types and also qualitatively analyzed the performances by taking few examples. We show that n-gram coattention encoders now capture the local context very well and also show the delimitation of coattention mechanism.

In the future, we would like to perform more deeper analysis on delimitations of coattention mechanism. Apart from this, our future line of research would be as follows: Incorporating the handcrafted features such as BM25 and study the performance. It would be interesting to see how the performance of the models will change with respect to the context based embeddings such as ELMo, BERT, etc.,

References

Chaitanya Sai Alaparthi. 2019. Microsoft ai challenge india 2018: Learning to rank passages for web question answering with deep attention networks. *arXiv preprint arXiv:1906.06056*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 126–134.

Sebastian Hofstätter, Navid Rekabsaz, Carsten Eickhoff, and Allan Hanbury. 2019. On the effect of low-frequency terms on neural-ir models. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1137–1140.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard De Melo. 2018. Co-pacrr: A context-aware neural ir model for ad-hoc retrieval. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 279–287.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *Advances in neural information processing systems*, pages 289–297.

Zhengdong Lu and Hang Li. 2013. A deep architecture for matching short texts. In *Advances in neural information processing systems*, pages 1367–1375.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Bhaskar Mitra and Nick Craswell. 2019. An updated duet model for passage re-ranking. *arXiv preprint arXiv:1903.07666*.

Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed

representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1291–1299.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *thirtieth AAAI conference on artificial intelligence*.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with bert. *arXiv preprint arXiv:1910.14424*.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. *arXiv preprint arXiv:1602.06359*.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd international conference on world wide web*, pages 373–374.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *International conference on machine learning*, pages 2397–2406.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR conference on research and development in information retrieval*, pages 55–64.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020. Repbert: Contextualized text embeddings for first-stage retrieval. *arXiv preprint arXiv:2006.15498*.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.

Language Model Metrics and Procrustes Analysis for Improved Vector Transformation of NLP Embeddings

Thomas Conley
University of Colorado
Colorado Springs
1420 Austin Bluffs Pkwy
Colorado Springs, CO, USA
tconley@uccs.edu

Jugal Kalita
University of Colorado
Colorado Springs
1420 Austin Bluffs Pkwy
Colorado Springs, CO, USA
jkalita@uccs.edu

Abstract

Artificial Neural networks are mathematical models at their core. This truism presents some fundamental difficulty when networks are tasked with Natural Language Processing. A key problem lies in measuring the similarity or distance among vectors in NLP embedding space, since the mathematical concept of distance does not always agree with the linguistic concept. We suggest that the best way to measure linguistic distance among vectors is by employing the Language Model (LM) that created them. We introduce Language Model Distance (LMD) for measuring accuracy of vector transformations based on the Distributional Hypothesis (*LMD_Accuracy*). We show the efficacy of this metric by applying it to a simple neural network learning the Procrustes algorithm for bilingual word mapping.

1 Introduction

The Distributional Hypothesis (Firth, 1961) inspired the development of embeddings that capture the meaning of language based on how words co-occur with each other (Mikolov et al., 2013a). Natural Language Processing relies heavily on these high dimensional vectors to represent words, phrases, sentences or documents, in a form that can be processed by deep neural networks which were originally designed for tasks related to computer vision. Input embeddings are transformed by network layers into output vectors which represent solutions to many NLP tasks (Ruder et al., 2019).

In order to learn these transformations, a network must be able to calculate the difference between predicted vectors and actual word vectors. This distance calculation is a crucial part of measuring loss, and performing back-propagation. These core functions of neural networks have primarily relied on mathematical processes without regard to linguistic principles. We demonstrate that NLP embedding transformation is better measured using linguistic similarity functions rooted in knowledge of languages rather than concepts such as Euclidean or angular distance, which assumes vectors to be "physical" objects.

1.1 Procrustes Analysis

Matrix transformation of vector spaces has been accomplished using Generalized Procrustes Analysis (GPA), ever since a computationally viable solution was devised (Gower, 1975). In particular, GPA has been used to great effect in geo-spatial shape manipulation (Duta, 2015; Crosilla et al., 2019) and qualitative data analysis (Maurício et al., 2016).

Shapes are represented by a series of landmark points in 2 or 3 dimensions. And in survey research, qualitative opinion data is represented by a Likert scale (Likert, 1932), occupying low dimensional space. In both cases, the vector spaces must be realigned and resized for meaningful comparison. Although these fields seem to differ, they use data structures that share characteristics with Natural Language Processing.

The orthogonal Procrustes algorithm produces an optimal transformation matrix R for mapping one vector space to another and appears to be useful in converting vector spaces for NLP tasks such as bilingual word mapping (Kementchedjhieva et al., 2018).

1.2 Procrustes Analysis for NLP tasks

Can a neural network learn to do Procrustes transformation? The answer, yes, should be non-controversial, since every neural network performs tensor transformation of input to output. However, tasks which require nuanced understanding of the meaning of words, such bilingual word mapping, are particularly difficult. Although there is some success when massive amounts of text are available for training, the problem is more acute when

Proceedings of the 17th International Conference on Natural Language Processing, pages 170–174
Patna, India, December 18 - 21, 2020.

resources for learning are scarce, as in machine translation of under resourced languages.

The difficulty with vector transformations in NLP is based on the nature of the data. NLP transformations by neural networks use distance measurements designed to work in L_p *space*. This implies numerical data. We show that such calculations of distance and accuracy are not as effective as measurements based on language models.

1.3 Image data and language data

We consider image data as raw data with physical dimensionality where, each dimension in a vector can be considered similar in measurement and meaning. As such, this data occupies L_p *space*; where vectors can be added together or multiplied by scalars without loss of their inherent meaning. For example, a vector representing a pixel is measured the same way, and has the same meaning, regardless of where it is in the image.

Thus, distance measurement among image vectors can use L_p *norm* or trigonometric calculations such as cosine distance. One specific kind of euclidean distance measurement is called *Procrustes Distance* and is the basis of Procrustes Analysis (Crosilla et al., 2019).

In NLP, distance measurement is less meaningful when it is based on Euclidean axioms rather than linguistic principles. Distance is the basis of error calculations and back-propagation, and so, the ability to calculate the derivative of these functions is essential for classic stochastic gradient descent (SGD) which is employed by neural networks today. Although there has been some research in non-differentiable losses (Engilberge et al., 2019) the mathematical requirements for these functions are not always suitable for NLP.

As opposed to raw data, feature data consists of vectors in which each dimension may have disparate meaning and measurement. Feature data does not exist in L_p *space*, and therefore measures of distance that rely on L_p *norm* or trigonometric calculations may not be meaningful. We consider NLP embeddings to be feature data, although they share some characteristics with raw data.

2 Language Models and Data

As in raw data, NLP vectors dimensions typically share values that are treated similarly and are thus undifferentiated in a sense. This seems to contradict the assertion that each NLP embedding dimension has a specific unique meaning like feature data. Instead, the meaning of a dimension is more like probability, representing how often a word is used with a particular meaning, rather than the actual meaning of the word.

Vectors with dimensions that differ in meaning, as in NLP embeddings, cannot be used with typical spatial measurements such as L_p *norm* and *cosine distance*. We contend that NLP vector distance can best be measured by the language models which represent the vectors. Therefore, we seek to replace mathematic calculations with predictions from language models. We simply rely on the language model itself to provide a distance measurement for our custom metric.

In this research, we use the Word2Vec model (Mikolov et al., 2013b) to produce a custom bilingual word mapping dataset. This dataset, combined with the GenSim model of keyed vectors (Řehůřek and Sojka, 2010), provides a distributional distance measurement based on word movers distance (Kusner et al., 2015).

Our neural network is a simple Multilayer Perceptron (MLP) which accepts Spanish word vectors as input and predicts English word vectors. This simple model was chosen because it is analogous to any layer found in innumerable, more complex, neural networks. Showing improved efficacy in this model should demonstrate improvement in any NLP task.

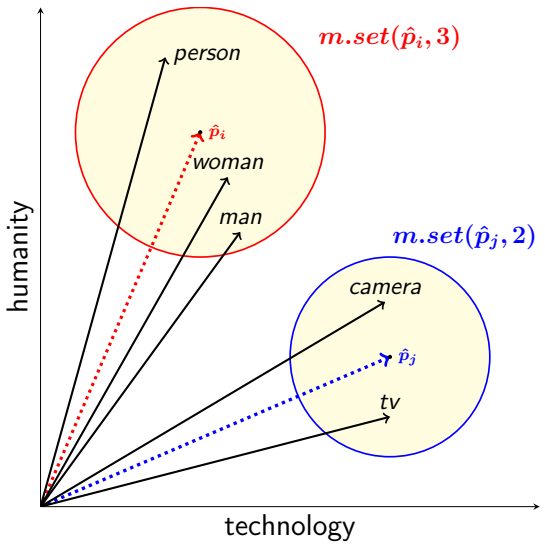


Figure 1: Illustration of the Distributional Hypothesis and Language Model Distance. The accuracy of predicted vectors $\hat{p_i}$ and $\hat{p_j}$, is based on membership in the set of $k = 2$ or $k = 3$ neighbors.

3 Language Model Distance

An exact measurement of equality is not possible for high-dimensional NLP embeddings. Embeddings of several hundred dimensions, and one-hot encoded vectors on the order of tens of thousands of dimensions, are particularly difficult to measure.

$$LMD(\hat{p}, t, m, k) = \begin{cases} True, & \text{if } t \in m.set(\hat{p}, k) \\ False, & \text{otherwise} \end{cases} \quad (1)$$

Instead, we suggest that the true measure of NLP vector distance is best provided by the model which defines the vectors. We present a family of metrics, Language Model Distance (LMD), which calculates distance and equality among NLP vectors by using the language model itself. LMD is defined as in Equation 1 where the distance between predicted vector $\hat{p}$ and known truth vector t, is provided by model m, given neighbor threshold k.

The distance measure is binary because it is based on set inclusion, and not physical or Euclidean distance. Thus, LMD can be used as a measure of accuracy, and records a true positive when t is within the neighborhood of the predicted vector ($t \in m.set(\hat{p}, k)$).

3.1 Measuring Accuracy with Language Model Distance

Figure 1 illustrates the distributional hypothesis by showing a simple clustering along 2 non-numeric dimensions. The circles represent neighborhoods $m.set(\hat{p}, k = 2)$ and $m.set(\hat{p}, k = 3)$. Note that the predicted vectors ($\hat{p}$) have no words directly associated with them, because no exact match is possible for floating point numeric vectors.

Thus we say that $LMD\_Accuracy(k)$ measures a positive result when truth vector (t) is within the k sized neighborhood of the predicted vector ($t \in m.set(\hat{p}, k)$). For example, $LMD\_Accuracy(3)$ measures the percentage of times that the true word answer was among the top 3 closest predicted words.

Distributional distance functions can be used in neural network metrics, loss, or activation functions, or used directly in similarity computation. However, inserting external language models into neural networks can be difficult as these networks are firmly rooted in mathematics which is not compatible with linguistic processes.

We solve these difficulties by defining a simple class shown in Figure 2. By including the language model as a static member of the class, methods of the class may be used as network internal functions with access to external language models.

```
1:  class DISTRIBUTION
2:      model ← Target Language Model
3:      method ACCURACY
4:          y_pred ← predicted vectors
5:          y_true ← known true vectors
6:          thresh ← neighbor threshold
7:          for each pred ∈ y_pred do
8:              y_neighbors ← model.closest(pred, thresh)
9:              if pred ∈ y_neighbors then
10:                 return True
11:             end if
12:         end for
13:         return False
14:     end method
15: end class
```

Figure 2: Implementation of Distributional Accuracy based on Language Model Distance. A static language model (line 2) allows linguistic functionality to be included in purely mathematical models.

4 Learning Orthogonal Procrustes Analysis

The Orthogonal Procrustes Algorithm is a process for finding the optimal mapping of one set of vectors to another. Typically, the vectors represent points in 2 or 3 dimensional space, for image processing, or they represent qualitative data measured in few dimensions (Maurício et al., 2016). After resizing and repositioning of vectors, an optimal rotation matrix R is produced by a method similar to singular value decomposition.

This classic approach to vector transformation has been explored as a solution for some NLP tasks (Sen et al., 2019; Kim et al., 2019). Therefore we ask: Can a neural network be trained to perform the same optimal transformation for NLP embeddings which occupy a much higher dimensional space?

Our task is to train a simple MLP to learn the optimal mapping R, between two disparate vector spaces representing a bilingual dataset. We measure the success of this task using LMD as the basis for accuracy as in Figure 2 and Equation 1.

We create two separate language models from a parallel corpus of European Parliament translations, the so called EuroParl dataset (Koehn, 2005). We use the Word2Vec model in continuous bag-of-words (CBOW) mode (Mikolov et al., 2013a) to build two separate distributions. By using a bilingual corpus, and training language models sep-

arately, we ensure that the models share a common domain, but the vector spaces remain separate. For training, we then map word vectors from one distribution to the other, using a set of 1000 most common words pairs, obtained from from a language learning website[1].

4.1 Results

Our results show that Orthogonal Procrustes Analysis can be learned for multilingual mapping of word vectors. Furthermore, Figure 3 demonstrates that LMD is effective as a basis for measuring the accuracy of this task.

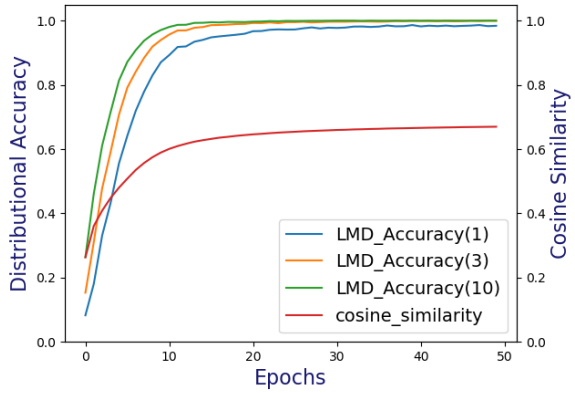


Figure 3: Results of Learning Orthogonal Procrustes Analysis showing a better measure of exact matches with *LMD_Accuracy* than with *cosine similarity*.

Figure 3 indicates that *LMD_Accuracy* is better at measuring similarity in NLP embeddings than *cosine similarity*. In this plot, *LMD_Accuracy(1)* indicates that the model exactly predicted the correct word in the output language. When *LMD_Accuracy(1)* is near 100% the value of *cosine similarity* should be near 1 which would indicate an exact match. The fact that *cosine similarity* cannot measure this exact match shows a weakness in this purely mathematical measurement compared with our language model-based measurement.

5 Learning General Procrustes Analysis

To further test, we try to learn General Procrustes Analysis; a much harder task because it requires the network to generalize.

We have just shown that a simple neural network can learn to transform vectors. This is non-controversial since all neural networks perform this task at every layer. However, not all networks are able to generalize. Using the same network configuration as before, we now evaluate embeddings that we have not seen in training, as is common. This is equivalent to learning the *Generalized Procrustes Algorithm*.

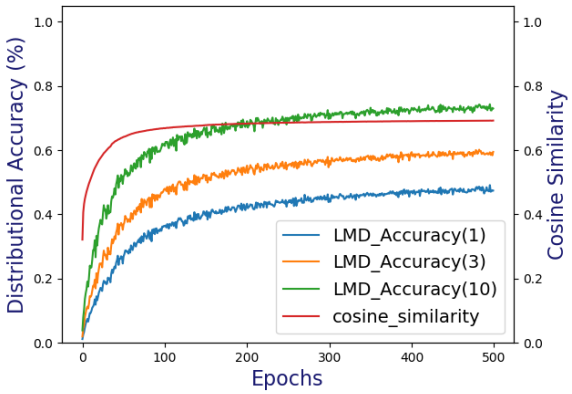


Figure 4: Results of Learning General Procrustes Analysis showing a comparable measure of exact matches between *LMD_Accuracy* and *cosine similarity*, when generalization is required

.

Results in Figure 4 show that *LMD_Accuracy* is more like cosine distance when generalization is required. Note that we use *LMD_Accuracy* only for metrics. This model uses $cosine\ similarity$ for error calculation and back-propagation. We conclude that such $L_p\ norm$ measurements can only drive generalization as far as they are able to measure accuracy.

The local variation in *LMD_Accuracy*, evident in Figure 4, may be significant as it may make determining the derivative of the function difficult. The derivative of *LMD_Accuracy* must be worked out before it can be incorporated into a loss function and be used in back-propagation. The overall shape of the curve, despite irregularities is encouraging as the slope may be computed using ordinary least squares in a calculation of rolling regression, or by other numerical methods.

6 Conclusion

We suggest that language model metrics described here may be incorporated directly into activation and loss functions, and may be used as an error measurement for back-propagation. We suggest this basic enhancement would improve the Generalized Procrustes Algorithm and other NLP processing in general. This is left for future work.

[1] http://www.englishnspanish.com

References

Fabio Crosilla, Alberto Beinat, Andrea Fusiello, Eleonora Maset, and Domenico Visintini. 2019. Orthogonal procrustes analysis. In *Advanced Procrustes Analysis Models in Photogrammetric Computer Vision*, pages 7–28. Springer.

Nicolae Duta. 2015. *Procrustes Shape Distance*, pages 1278–1279. Springer US, Boston, MA.

Martin Engilberge, Louis Chevallier, Patrick Pérez, and Matthieu Cord. 2019. Sodeep: a sorting deep net to learn ranking loss surrogates. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10792–10801.

John Rupert Firth. 1961. *Papers in Linguistics 1934-1951: Repr*. Oxford University Press.

John C Gower. 1975. Generalized procrustes analysis. *Psychometrika*, 40(1):33–51.

Yova Radoslavova Kementchedjhieva, Sebastian Ruder, Ryan Cotterell, and Anders Søgaard. 2018. Generalizing procrustes analysis for better bilingual dictionary induction. In *22nd Conference on Computational Natural Language Learning (CoNLL 2018)*, pages 211–220. Association for Computational Linguistics.

Yunsu Kim, Petre Petrov, Pavel Petrushkov, Shahram Khadivi, and Hermann Ney. 2019. Pivot-based transfer learning for neural machine translation between non-english languages. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 865–875.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86. Citeseer.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. volume 37 of *Proceedings of Machine Learning Research*, pages 957–966, Lille, France. PMLR.

Rensis Likert. 1932. A technique for the measurement of attitudes. *Archives of psychology*.

Angélica Maurício, A.B. Palazzo, Valeria Caselato, and Helena Bolini. 2016. Generalized procrustes analysis and external preference map used to consumer drivers of diet gluten free product. *Food and Nutrition Sciences*, 07:711–723.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2019. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*, 65:569–631.

Sukanta Sen, Kamal Kumar Gupta, Asif Ekbal, and Pushpak Bhattacharyya. 2019. Multilingual unsupervised nmt using shared encoder and language-specific decoders. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3083–3089.

Cognitively Aided Zero-Shot Automatic Essay Grading

Sandeep Mathias[1], **Rudra Murthy**[1,2], **Diptesh Kanojia**[1,3], and **Pushpak Bhattacharyya**[1]
[1] Department of Computer Science & Engineering, IIT Bombay
[2] IBM Research India Limited
[3] IITB-Monash Research Academy
{sam,diptesh,pb}@cse.iitb.ac.in, rmurthyv@in.ibm.com

Abstract

Automatic essay grading (AEG) is a process in which machines assign a grade to an essay written in response to a topic, called the prompt. Zero-shot AEG is when we train a system to grade essays written to a new prompt which was not present in our training data. In this paper, we describe a solution to the problem of zero-shot automatic essay grading, using cognitive information, in the form of gaze behaviour. Our experiments show that using gaze behaviour helps in improving the performance of AEG systems, especially when we provide a new essay written in response to a new prompt for scoring, by an average of almost **5 percentage points** of QWK.

1 Introduction

One of the major challenges in machine learning is the requirement of a large amount of training data. AEG systems perform at their best when they are trained in a prompt-specific manner - i.e. the essays that they are tested on are written in response to the **same** prompt as the essays they are trained on (Zesch et al., 2015). These systems perform badly when they are tested against essays written in response to a different prompt.

Zero-shot AEG is when our AEG system is used to grade essays written in response to a completely different prompt. In order to solve this challenge of lack of training data, we use cognitive information learnt by gaze behaviour of readers to augment our training data and improve our model.

Automatic essay grading has been around for over half a century ever since Page (1966)'s work (Beigman Klebanov and Madnani, 2020). While there have been a number of commercial systems like E-Rater (Attali and Burstein, 2006) from the Educational Testing Service (ETS), most modern-day systems use deep learning and neural networks, like convolutional neural networks (Dong and Zhang, 2016), recurrent neural networks (Taghipour and Ng, 2016), or both (Dong and Zhang, 2016). However, all these systems rely on the fact that their training and testing data is from the same prompt.

Quite often, at run time, we may not have essays written in response to our target prompt (i.e. the prompt which our essay is written in response to). Because of the lack of training data, especially when training a model for essays written for a new prompt, many systems may fail at run time. To solve this problem, we propose a multi-task approach, similar to Mathias et al. (2020), where we learn a reader's gaze behaviour for helping our system grade new essays.

In this paper, we look at a similar approach proposed by Mathias et al. (2020) to grade essays using cognitive information, which is learnt as an auxiliary task in a multi-task learning approach. Multi-task learning is a machine-learning approach, where the model tries to solve one or more auxiliary tasks to solve a primary task (Caruana, 1998). Similar to Mathias et al. (2020), the scoring of the essay is the primary task, while learning the gaze behaviour is the auxiliary task.

Contribution. In this paper, we describe a relatively new problem - zero-shot automatic essay grading - and propose a solution for it using gaze behaviour data. We show a **5 percentage points** increase in performance when learning gaze behaviour, as opposed to without using it.

1.1 Gaze Behaviour Terminology

We use the following gaze behaviour terms as defined by Mathias et al. (2020). An ***Interest Area*** (IA) is a part of the screen that is of interest to us. These areas are where some text is displayed, and not the background on the left / right, as well as above / below the text. **Each word** is a separate

Proceedings of the 17th International Conference on Natural Language Processing, pages 175–180
Patna, India, December 18 - 21, 2020.

and unique IA. A ***Fixation*** is an event when the reader's eye fixates on a part of the screen. We are only concerned with fixations that occur inside interest areas. The fixations that occur in the background are ignored. ***Saccades*** are eye movements as the eye moves from one fixation point to the next. ***Regressions*** are a type of saccade where the reader moves from the current interest area to an *earlier* one.

1.2 Organization of the Paper

The rest of the paper is organized as follows. Section 2 describes the motivation for our work. Section 3 describes some of the related work in the area of automatic essay grading. Section 4 describes the essay dataset, as well as the gaze behaviour dataset. Section 5 describes our experiment setup. We report our results and analyze them in Section 6 and conclude our paper in Section 7.

2 Motivation

As stated earlier, in Section 1, one of the challenges for machine-learning systems is the requirement of training data. Quite often, we may not have training data for an essay, especially if the essay is written in response to a new prompt. Without any labeled data, in the form of scored essays, we cannot train a system properly to grade the essays.

Zero-shot automatic essay grading is a way in which we overcome this problem. In zero-shot automatic essay grading, we train our system on essays written to different prompts, and test it on essays written in response to the target prompt. One drawback of this approach is that it would not be able to use the properties of the target essay set in training the model. Therefore, as a way to alleviate this problem, we learn cognitive information, in the form of gaze behaviour, for the essays to help our automatic essay grading system grade the essays better.

3 Related Work

While there has been work done on developing systems for automatic essay grading, all of them describe systems which use some of the essays the system is tested on as part of the training data (as well as validation data, where applicable) (Chen and He, 2013; Phandi et al., 2015; Taghipour and Ng, 2016; Dong and Zhang, 2016; Dong et al., 2017; Zhang and Litman, 2018; Cozma et al., 2018; Tay et al., 2018; Mathias et al., 2020).

One of the solutions to solve the problem was using cross-domain AEG, where systems were trained using essays in a set of source prompt / prompts and tested on essays written in response to the target prompt. Some of the work done to study cross-domain AEG were Zesch et al. (2015) (who used task-independent features), Phandi et al. (2015) (who used domain adaptation), Dong and Zhang (2016) (who used a hierarchical CNN layers) and Cozma et al. (2018) (who used string kernels and super word embeddings). In all of their works, they defined a ***source prompt*** which is used for training and a ***target prompt*** which is used for validation and testing.

To the best of our knowledge, we are the first to explore the task of ***Zero-shot*** automatic essay grading, as a way to alleviate the challenge of a lack of graded essays (written in response to the target prompt) for an automatic essay grading system. In our approach, **we do not use the target prompt essays even for validation**, thereby making it truly zero-shot.

4 Datasets

In this section, we discuss our essay grading dataset and the gaze behaviour dataset which we used.

4.1 Essay Dataset Details

For our experiments, we use the Automatic Student Assessment Prize (ASAP)'s AEG dataset[1]. This dataset is one of the most widely-used essay grading datasets, consisting of 12,978 graded essays, written in response to 8 essay prompts. The prompts are either argumentative, narrative, and source dependent responses. Details of the dataset are summarized in Table 1.

4.2 Gaze Behaviour Dataset

For our experiments, we use the same essay grading dataset as Mathias et al. (2020). We use 5 attributes of gaze behaviour, namely dwell time (the total time that the eye has fixated on a word), first fixation duration (the duration of the first fixation of the reader on a particular word), IsRegression (whether or not there was a regression from a particular interest area or not), Run Count (the number of times an interest area was fixated on), and Skip (whether or not the interest area was skipped).

[1]The dataset can by downloaded from https://www.kaggle.com/c/asap-aes/data.

Prompt ID	Number of Essays	Score Range	Mean Word Count	Essay Type
Prompt 1	1783	2-12	350	Persuasive
Prompt 2	1800	1-6	350	Persuasive
Prompt 3	1726	0-3	150	Source-Dependent
Prompt 4	1770	0-3	150	Source-Dependent
Prompt 5	1805	0-4	150	Source-Dependent
Prompt 6	1800	0-4	150	Source-Dependent
Prompt 7	1569	0-30	250	Narrative
Prompt 8	723	0-60	650	Narrative
Total	12976	0-60	250	–

Table 1: Statistics of the 8 prompts from the ASAP AEG dataset.

Essay Set	0	1	2	3	4	Total
Prompt 3	2	4	5	1	N/A	12
Prompt 4	2	3	4	3	N/A	12
Prompt 5	2	1	3	5	1	12
Prompt 6	2	2	3	4	1	12
Total	8	10	15	13	2	48

Table 2: Number of essays for each essay set which we collected gaze behaviour, scored between 0 to 3 (or 4).

The gaze behaviour was collected from 8 different annotators, who read only 48 essays (out of the almost 13,000 essays in the ASAP AEG dataset) from the source dependent response essay sets. Table 2 summarizes the distribution of essays across the different essay sets that we collect gaze behaviour data for.

Table 3 gives the details of the different annotators used by Mathias et al. (2020). We evaluated the annotator's performance on 3 different metrics - QWK, Close and Correct. **QWK** is the Quadratic Weighted Kappa agreement (Cohen, 1968) between the score given by the annotator and the ground truth score from the dataset. **Correct** is the number of times (out of 48) that the annotator **exactly** agreed with the ground truth score, and **Close** is the number of times (out of 48) where the annotator disagreed with the ground truth score by **at most 1 score point**.

More details about the dataset and its creation are found in Mathias et al. (2020).

5 Experiment Setup

In this section, we describe our experiment setup, such as the evaluation metric, network architecture and hyperparameters, etc.

5.1 Evaluation Metric

For evaluating our system, we use Cohen's Kappa with Quadratic Weights, i.e. Quadratic Weighted Kappa (QWK) (Cohen, 1968). This evaluation metric is most frequently used for automatic essay grading experiments because it is sensitive to differences in scores, and takes into account chance agreements (Mathias et al., 2018).

5.2 Network Architecture

Figure 1 shows the architecture of our system. The essay is split into different sentences and each sentence is tokenized and given as input at the Embedding Layer. In this layer, for each token, we output the corresponding word embedding, which is given as input to the next layer - the Word-level CNN layer.

The Word-level CNN layer learns local representations of nearby words, as well as the gaze behaviour. The outputs of the word-level CNN layer are then pooled at the word-level pooling layer to get a sentence representation for each sentence.

Each sentence representation is then sent through an LSTM (Hochreiter and Schmidhuber, 1997) layer, whose output is pooled through a sentence-level attention layer, to get the essay representation.

The essay representation from the sentence-level attention layer is then sent through a Dense layer, from which we learn the essay scores. For both the tasks (learning gaze behaviour, as well as scoring the essay), we minimize the mean squared error loss.

ID	Sex	Age	Occupation	TA?	L1 Language	English Score	QWK	Correct	Close
Annotator 1	Male	23	Masters student	Yes	Hindi	94%	0.611	19	41
Annotator 2	Male	18	Undergraduate	Yes	Marathi	95%	0.587	24	41
Annotator 3	Male	31	Research scholar	Yes	Marathi	85%	0.659	21	43
Annotator 4	Male	28	Software engineer	Yes	English	96%	0.659	26	44
Annotator 5	Male	30	Research scholar	Yes	Gujarati	92%	0.600	19	42
Annotator 6	Female	22	Masters student	Yes	Marathi	95%	0.548	19	40
Annotator 7	Male	19	Undergraduate	Yes	Marathi	93%	0.732	21	46
Annotator 8	Male	28	Masters student	Yes	Gujarati	94%	0.768	29	45

Table 3: Profile of the annotators

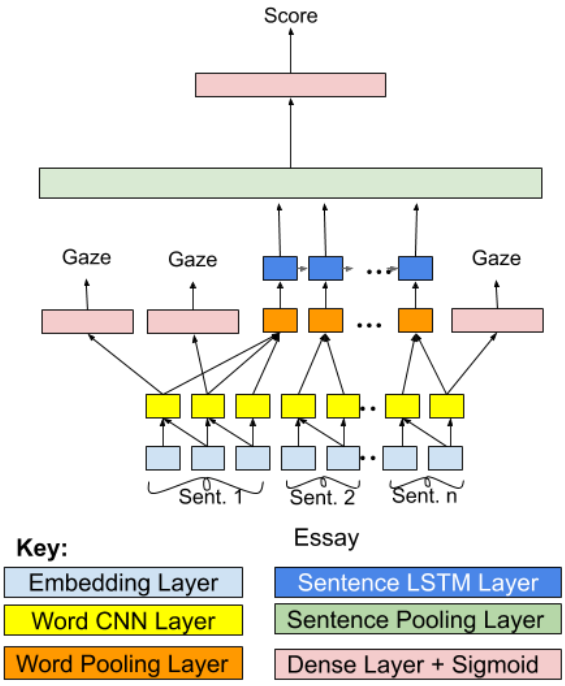


Figure 1: Architecture of our gaze behaviour system, showing an input essay of n sentences, with the outputs being the gaze behaviour (whenever applicable), and the overall essay score.

5.3 Network Hyperparameters

We use the **50 dimension** GloVe pre-trained word embeddings (Pennington et al., 2014). We run our experiments over a **batch size of 200**, for **50 epochs**. We set the **learning rate as 0.001**, and the **dropout rate as 0.5**. The word-level CNN layer has a **kernel size of 5**, with **100 filters**. The sentence-level LSTM layer has **100 hidden units**. We use the RMSProp Optimizer (Dauphin et al., 2015) with an **initial learning rate of 0.001** and **momentum of 0.9**. Along with the network hyperparameters, we also weigh the loss functions of the different gaze behaviour attributes differently, using the same weights as Mathias et al. (2020), namely **0.05 for DT and FFD**, **0.01 for IR and RC**, and **0.1 for Skip**.

5.4 Normalization and Binning

While training our model, we scale the essay scores for all the data (training, testing and validation) to a range of $[0, 1]$. For calculating the final scores, as well as the QWK, we rescale the predictions of the essay score back to the score range of the essays.

We also bin the gaze behaviour attributes as described in Mathias et al. (2020). Binning is done to take into account the idiosyncracies of the gaze behaviour of individual readers (i.e. some people may read faster, others slower, etc.). Whenever we use gaze behaviour, we scale the value of the gaze behaviour bins to the range of $[0, 1]$ as well.

5.5 Experiment Configurations

We run our experiments in the following configurations. **No Gaze** is a single-task learning experiment, where we only learn to score the essay. **Gaze** is the multi-task learning approach, where we learn gaze behaviour as an auxiliary task, and score the essay as the primary task.

5.6 Evaluation Method

We use **five-fold cross-validation** to evaluate our system. For each fold, the testing data consists of essays from the target prompt and the training data and validation data comprise of essays from the other 7 prompts.

6 Results and Analysis

Table 4 gives the results of our experiments. The results reported are on the target essay set for the mean of the 5 folds. For each fold, we record the performance of the model on the target essay set, corresponding to the epoch which had the best QWK for the development set. Table 4 reports the mean performance for all 5 folds.

From the table, we see that in most of the essay sets, we are able to see an improvement in perfor-

Target Essay Set	No Gaze	Gaze
Prompt 1	0.319	**0.423***
Prompt 2	0.391	**0.439***
Prompt 3	0.508	**0.545***
Prompt 4	0.548	**0.626***
Prompt 5	0.548	**0.628***
Prompt 6	0.599	**0.600**
Prompt 7	0.362	**0.420***
Prompt 8	**0.316**	0.286
Mean QWK	0.449	**0.498***

Table 4: Results of our experiments with and without using gaze behaviour. Improvements which are statistically significant (with $p < 0.05$), when gaze behaviour is used, are marked with a *

mance. In order to verify if the improvements were statistically significant, we use the 2-tailed Paired T-Test with a significance level of $p < 0.05$. Statistically significant improvements where we use gaze behaviour data are marked with a * next to the result.

Out of the 8 essay sets, the only essay set where the performance using gaze behaviour falls short compared to when we do not use gaze behaviour is in Prompt 8. One of the main reasons for this is that the essays in Prompt 8 are very long compared to the other essay sets. When they are absent from the training data, the system is unable to learn about the existence of long essays, which could also be the reason that those essays are scored badly.

7 Conclusion and Future Work

In this paper, we discussed an important problem for automatic essay grading, namely **zero-shot** automatic essay grading, where we have no labeled essays written in response to our target prompt, present at the time of training.

We showed that, by using gaze behaviour, we are able to learn cognitive information which can help improve our AEG system.

In the future, we plan to extend our work to other tasks, like grading of essay traits, using gaze behaviour.

References

Yigal Attali and Jill Burstein. 2006. Automated essay scoring with e-rater®v.2. *The Journal of Technology, Learning and Assessment (JTLA)*, 4(3).

Beata Beigman Klebanov and Nitin Madnani. 2020. Automated evaluation of writing – 50 years and counting. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7796–7810, Online. Association for Computational Linguistics.

Rich Caruana. 1998. *Multitask Learning*, pages 95–133. Springer US, Boston, MA.

Hongbo Chen and Ben He. 2013. Automated essay scoring by maximizing human-machine agreement. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1741–1752, Seattle, Washington, USA. Association for Computational Linguistics.

Jacob Cohen. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213.

Mădălina Cozma, Andrei Butnaru, and Radu Tudor Ionescu. 2018. Automated essay scoring with string kernels and word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 503–509, Melbourne, Australia. Association for Computational Linguistics.

Yann Dauphin, Harm De Vries, and Yoshua Bengio. 2015. Equilibrated adaptive learning rates for non-convex optimization. In *Advances in neural information processing systems*, pages 1504–1512.

Fei Dong and Yue Zhang. 2016. Automatic features for essay scoring – an empirical study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1072–1077, Austin, Texas. Association for Computational Linguistics.

Fei Dong, Yue Zhang, and Jie Yang. 2017. Attention-based recurrent convolutional neural network for automatic essay scoring. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 153–162, Vancouver, Canada. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Sandeep Mathias, Diptesh Kanojia, Kevin Patel, Samarth Agrawal, Abhijit Mishra, and Pushpak Bhattacharyya. 2018. Eyes are the windows to the soul: Predicting the rating of text quality using gaze behaviour. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2352–2362, Melbourne, Australia. Association for Computational Linguistics.

Sandeep Mathias, Rudra Murthy, Diptesh Kanojia, Abhijit Mishra, and Pushpak Bhattacharyya. 2020. Happy are those who grade without seeing: A multi-task learning approach to grade essays using gaze behaviour. In *Proceedings of the 1st Conference of the*

Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, pages 858–872, Suzhou, China. Association for Computational Linguistics.

Ellis B Page. 1966. The imminence of... grading essays by computer. *The Phi Delta Kappan*, 47(5):238–243.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Peter Phandi, Kian Ming A. Chai, and Hwee Tou Ng. 2015. Flexible domain adaptation for automated essay scoring using correlated linear regression. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 431–439, Lisbon, Portugal. Association for Computational Linguistics.

Kaveh Taghipour and Hwee Tou Ng. 2016. A neural approach to automated essay scoring. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1882–1891, Austin, Texas. Association for Computational Linguistics.

Yi Tay, Minh Phan, Luu Anh Tuan, and Siu Cheung Hui. 2018. Skipflow: Incorporating neural coherence features for end-to-end automatic text scoring.

Torsten Zesch, Michael Wojatzki, and Dirk Scholten-Akoun. 2015. Task-independent features for automated essay grading. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 224–232, Denver, Colorado. Association for Computational Linguistics.

Haoran Zhang and Diane Litman. 2018. Co-attention based neural network for source-dependent essay scoring. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 399–409, New Orleans, Louisiana. Association for Computational Linguistics.

Automated Arabic Essay Evaluation

Abeer Alqahtani
Department of Computer Science
Al Immam Mohammad Ibn Saud
Islamic University(IMSIU)
Riyadh, Saudi Arabia
aakalqahtani@sm.imamu.edu.sa

Amal Alsaif
Department of Computer Science
Al Immam Mohammad Ibn Saud
Islamic University(IMSIU)
Riyadh, Saudi Arabia
asmalsaif@imamu.edu.sa

Abstract

Although the manual evaluation of essays is a time-consuming process, writing essays has a significant role in assessing learning outcomes. Therefore, automated essay evaluation represents a solution, especially for schools, universities, and testing companies. Moreover, the existence of such systems overcomes some factors that influence manual evaluation such as the evaluator's mental state, the disparity between evaluators, and others. In this paper, we propose an Arabic essay evaluation system based on a support vector regression (SVR) model along with a wide range of features including morphological, syntactic, semantic, and discourse features. The system evaluates essays according to five criteria: spelling, essay structure, coherence level, style, and punctuation marks, without the need for domain-representative essays (a model essay). A specific model is developed for each criterion; thus, the overall evaluation of the essay is a combination of the previous criteria results. We develop our dataset based on essays written by university students and journalists whose native language is Arabic. The dataset is then evaluated by experts. The experimental results show that 96% of our dataset is correctly evaluated in the overall score and the correlation between the system and the experts' evaluation is 0.87. Additionally, the system shows variant results in evaluating criteria separately.

1 Introduction

Automated essay scoring (AES) or so-called automated essay evaluation (AEE) systems came to facilitate the evaluation task of students' writings. Assigning questions that demonstrate writing skills, such as linguistic skills and creativity, is crucial. However, the evaluation process is laborious, particularly with a large number of essays, such as during examination boards.

AES systems assist essay authors, editorial boards, publishers, and newspaper editors by overcoming some of the shortcomings of traditional evaluation. For instance, AES systems reduce variabilities between instructors' viewpoints or biases resulting from a good point in essay that causes an evaluator to ignore other mistakes (Janda et al., 2019). Moreover, AES systems are tools that can assist both new instructors and students for training and improving writing skills.

For the English language, there have been many studies drafted on AES in addition to the development of commercial applications used in English-learning institutes. In comparison, AES systems for Arabic seem restricted to short-answer questions with predefined answer models from instructors. This limitation stems from complexities of the Arabic language and a lack of Arabic natural language processing (NLP) resources.

In this paper, we attempt to fill this gap by providing an Arabic essay evaluation system using a machine learning method that does not require a model essay.

2 Related Work

In English, there are several related efforts. For example, the project essay grader (PEG) is one of the earliest scoring systems. The system was based on a statistical method to predict the score. PEG

Proceedings of the 17th International Conference on Natural Language Processing, pages 181–190
Patna, India, December 18 - 21, 2020.

succeeded in predicting the surface structure of an essay, but it did not meet the semantic criteria (Chung & O'Neil, 1997; Kukich, 2000).

In the intelligent essay assessor (IEA), latent semantic analysis (LSA) is used to evaluate essay content. In addition, IEA can be used to evaluate writing style and detect plagiarism (Dessus and Lemaire, 2001). The E-rater (Burstein, 2003) and IntelliMetric (Elliot, 2003) are evaluation systems that rely on linguistic features extracted by NLP techniques to evaluate common criteria.

In contrast to previous systems, the Bayesian essay test scoring system (BETSY) is non-commercial and can be used for research. BETSY uses Bayesian models to evaluate essays for content and style (Dikli, 2006).

Although commercial systems have restrictions to accessing system details, AES systems have attracted the attention of many researchers. Most studies, such as Surya et al. (2018), have focused on the automatic student assessment prize (ASAP) dataset.

Surya et al. (2018) used machine learning with nine surfaces, deep features, and three algorithms including support vector machine (SVM), k-nearest neighbors (kNN), and linear regression. The obtained accuracy ranges from 73% to 93% according to the dataset class and the algorithms used. In addition, Alikaniotis, Yannakoudakis, and Rei (2016) and Dong and Zhang (2016) have employed deep-learning algorithms and obtained encouraging results.

In Arabic, there are several studies for short-answer questions scoring. Nahar & Alsmadi (2009) based on light stemming and assigning weights over words in the model answer. For the system to consider semantics, the instructor must manually attach the synonyms. In comparison, Gomaa and Fahmy (2014) combined similarity measures, such as string, corpus, and knowledge-based similarity for 610 Arabic short answers, which were translated to English due to the lack of Arabic resources. However, this approach requires great effort for translation and then scoring.

Apart from semantics, Al-Shalabi (2016) proposed a scoring system for online exams using stemming and Levenshtein string-based similarity measures. Shehab, Faroun, and Rashad (2018) conducted a comparison between several string-based (Damerau–Levenshtein and N-gram) and corpus-based similarity measures (LSA and DISCO) on 210 answers. They found that applying N-gram with removing stop words produced the best results.

Other than short-answer scoring, we only came across three studies for essay scoring (Alghamdi et al., 2014; Azmi, Al-Jouie, & Hussain, 2019; Alqahtani & Alsaif, 2019).

First, Alghamdi et al. (2014) conducted a study based on a linear regression algorithm. They used LSA, the number of words, and spelling mistakes to predict an essay's score. The proposed system was applied to 579 essays collected from undergraduate university students and scored by two instructors. The results showed that 96.72% of essays were scored correctly, while the correlation between the system and manual scoring was 0.78, which is close to the value of 0.7 obtained by inter-human correlation.

Second, Azmi, Al-Jouie, and Hussain (2019) collected 360 essays from students in middle and high schools. The essays were evaluated by two school instructors according to criteria obtained from a questionnaire given to instructors. The criteria can be presented as semantic analysis, writing style, and spelling mistakes. However, there was a difference in assigning each criterion weight, but the majority was 5%, 40%, and 10% for semantic analysis, writing style and spelling mistakes, respectively. The proposed system was based on LSA to evaluate semantics while rhetorical structure theory (RST) and other features were used to evaluate the writing style. Finally, the system employed AraComLex (Attia, Pecina, Toral, Tounsi, & Genabith, 2011) to detect spelling mistakes. The system achieved an accuracy of 90% while the correlation to the manual evaluation was 0.756, which outperformed the inter-human correlation of 0.709.

Both Alghamdi et al. (2014) and Azmi, Al-Jouie, and Hussain (2019) relied on the LSA approach, which requires domain-representative essays. Furthermore, outputs of Alghamdi et al. (2014) are given as an overall score without details about the score for each criterion. In contrast, Azmi, Al-Jouie, and Hussain (2019) employ rules to evaluate writing style and spelling mistakes separately. However, half of the assigned score is based on LSA, which in turn requires pre-defined models.

Recently, (Alqahtani & Alsaif, 2019) proposed a rule-based system to evaluate Arabic essays without the need to train on domain-representative essays. They adopted an evaluation criteria scheme

based on Arabic literary resources and university instructors' experiences as following: spelling, grammar, structure, cohesion, style and punctuation marks. Then, scheme was given to two experts to evaluate 100 essays for university students. The system follows a set of rules to evaluate the previous criteria based on some facts and analyses to evaluate the overall score beside the specific criteria scores. The system accuracy was 73% in the overall score while there were variations in evaluating criteria separately. Although this study does need a model essay, it is limited to set of rules that does not include semantic.

Therefore, in this paper, we propose an Arabic essay evaluation system does not need to train on predefined essays represent domain by using a machine learning algorithm and a wide range of features to evaluate the specific criteria besides the overall score.

3 Dataset

Our dataset can be classified into three parts: essays written by undergraduate/graduate Arabic native students with university-level; unedited essays written to be published in one of the Saudi newspapers and essays have been published in different newspapers. Any handwritten copies have been retyped by computer exactly as they were written. Altogether, our dataset contains 200 (MSA) essays with an approximately average length of 250 words (3KB) in several topics.

This dataset has been given to two Arabic experts hold master's degree in Arabic language to evaluate essays following the criteria and the evaluation rubric shown precisely in (Alqahtani & Alsaif, 2019) which include: spelling, grammar, structure, coherence and cohesion level, style and punctuation marks.

Therefore, in spelling criterion, evaluators concern about the correct spell of words; therefore, they detect each spelling mistake and classify it to one of four types: mistakes on *<hmzp/همزة/ء>*, replacement letters, extra letters and neglecting letters. For structure criterion, they check the existence of four essential parts; title, introduction, body and conclusion. In coherence, they evaluate the coherence between the title and remaining parts and check the cohesion between essay parts. This criterion also concerns with the correct use of connectives. To evaluate style of essay, they consider, words repetition, length of sentences, word choice and avoiding lengthy speech. In addition, they evaluate the punctuation marks according to the Arabic rules.

4 Automated Essay Evaluation

We intend to model Arabic essay evaluation based on the supervised linear model Support Vector Regression (SVR) (Awad & Khanna, 2015) along with different levels of features. We develop a specific model for each criterion. The overall evaluation of an essay will be a combination of the models' outputs. We follow this procedure to ensure that each criterion model takes the full advantage of the used features hence more accurate in the overall score. In addition, existence of separated model per criterion will assist in the future to provide a valuable feedback for the user. In the preprocessing step, we applied normalization, then stemming using Buckwalter stemmer (Buckwalter, 2004). Our features extracted by considering the criteria followed by humans. According to the system output, which is numeric scores, we try a wide range of features represented by numbers at different levels including:

A. Surface features

The surface features demonstrate only features dealing with the text itself such as word frequency. Each feature is followed by abbreviation for ease of reference as follows:

Essay Length (F1): is measured by number of tokens/words result of white space tokenization of essay.

Number of paragraphs (F3, F4) and sentences (F5): we considered each line as a paragraph so that when the writer moves to the next line because there is no space it is considered as one line (F3). Additionally, we checked if number of paragraphs is greater than one or not (F4). To count number of sentences (F5), we divided essays by period, comma and a list of connectives that usually used to connect two sentences in Arabic according to (Alsaif, 2012) study.

Words per essay parts (F6), (F7), (F8), (F9), and (F16): knowing the length of a specific part of essay, may lead to identifying its role in the essay, (e.g., when the first part is the shortest, that may

indicate it is the title). Generally, distributing features over essay parts lead to detect the effective features to evaluate a specific criterion. So, in separated features, we counted the number of words in first paragraph/title (F6), second paragraph /introduction (F7), last paragraph/conclusion (F8), and number of paragraphs in the middle/body (F9). Also, we checked if the first paragraph length is less than or equal to ten words (F16).

Average, maximum, and minimum length (F10–F15): in separated features, based on number of words we calculated the average length of sentences (F10), longest paragraph (F12) and the shortest paragraph (F13). Likewise, we calculated the longest sentence (F14) and the shortest sentence (F15).

Paragraph has a specific mark (F19–F22): for example, the presence of some marks may indicate the essence of paragraph. Separately, we checked each essay part if it contains parentheses, colon or question mark (F19–F22).

Number of <hmzp/همزة/ء> (F22) in essay: we counted words that contain *hmzp on AlOlf* /أ and إ, *hmzp on AlwAw* /ؤ, *hmzp on line*/ء and *hmzp on nbrp*/ئ (F22) to assist in spelling evaluation.

B. Syntactic and Morphological features

Parts of speech (POS) frequency in essay (F2–F99): by using the (POS) tag provided by the MADAMIRA analyzer (Pasha et al., 2014), we counted the amount of punctuation (F23), pronouns (F24), prepositions (F25), verbs (F26), nouns (F27), adjectives (F28), adverbs (F29) and numbers (F30) in the whole essay. In addition, we calculated these features for essay parts (paragraphs and sentences). Also, we checked paragraphs and sentences that start with a specific POS (adjectives and prepositions) separately (F31–F99).

Number of nominal and verbal sentence (F100) and (F101): for each sentence, we checked the first three words. If the sentence included a verb, it was considered as a verbal (F100), otherwise; it was considered as a nominal phrase (F101).

Spelling mistakes in the whole essay (F102–F109): In line with Alqahtani and Alsaif (2019), we used the FARASA spell checker (*FARASA: Advanced Tools for Arabic*, 2019) to classify the mistakes that FARASA provided into four classes: mistakes on *hmzp*, replacement letters, extra letters, and omission letters. Therefore, as features, we counted the number of spelling mistakes in any type of *hmzp* (F102), mistakes in replacement (F103), extra letters (F0104), or omissions (F105). Additionally, considering all the previous features, if its value was more than or equal to one, we assigned 1 to indicate a mistake; otherwise, we assigned 0 (F106–F 109).

Aljzm/الجزم particles (F110) and (F111): we counted number of *aljzm* particles in the essay as they cause a change in the subsequent verbs *(lm/*لم *lA AlnAhyp /*لا الناهية*, lA AlOmr /*لام الأمر*, lmA/*لما*)* (F110) as well as the number of times they were followed by a verb (F200). Also, we counted the number of cases of *aljzm* particles followed by a plural verb ending with *n/*ن, as this case affects the word form (F201) (Ali, 2019).

kAn wOxwAthA /Kana and her Sisters/كان وأخواتها (F112) and (F113): we counted the number of *kAn wOxwAthA*, which include: (*kAn/*كان, *ODHY /*أضحى*, mAzAl/*مازال, *lys/*ليس, *mAZl/*ماظل, *OmsY /*أمسى*, mAfti/*ماقتئ, *bAt/*بات, *SAr/*صار, *Zl/*ظل, *mAAnfk/*ماانفك, *mAbrH/*مابرح, *mAdAm/*مادام, *OSbH* أصبح) as they may affect the surrounding word forms (F112) (Ali, 2018).

In~ wOxwAthA /Inna and her Sisters/إن وأخواتها (F114): Moreover, we counted number of *In~ wOxwAthA*: (On~ /أن, In~/إن, kOn /كأن, lkn/لكن, lyt/ليت, lA/لا, lEl/لعل) in the essay for their effect on words forms (Ali, 2018).

Morphological mistakes (F117–F119): to count the number of words that were morphologically incorrect, we counted the number of words that could be analyzed by MADAMIRA (F117), the number of words that could not be analyzed by MADAMIRA (F118) and the number of words that their lemmas were not included in alWaseet or Contemporary dictionaries using SAFAR platform (*SAFAR: Software Architecture For ARabic*, 2013). In addition, we checked the style of plural words so, if the type of word was (p/plural) or the word ended with (ات) and its lemma ended with (ة/ه), but it does not belong to alWaseet or Contemporary dictionaries, then the number of mistakes in sound feminine plural increases (F119). These features may assist in evaluate spelling and style criteria.

C. Lexical features

Number of words without stop words (F123): refers to the number of words without stop words frequently used in Arabic text.

Introduction and conclusion keywords (F124–F127): usually, the introductory section may include some keywords used to pave a topic. Likewise, the writer may use specific words to conclude or summarize the essay. Therefore, we have two features, the first for checking if the first or second paragraph contains introductory keywords such as (*bdAyp/*بداية, *ntHdv/*نتحدث, *ntklm/*نتكلم, *nstErD/*نستعرض, or *AlmwDwE/*الموضوع, etc.) (F124). We did the same by checking the last paragraph for the conclusion (F125). The common words used to conclude in our dataset and Arabic essays generally include *(OrY/*أرى, *OxyrA/*أخيرا, *Orjw/*أرجو, *wjhp nZr/* وجهة نظر, *OqtrH/*أقترح, *OtmnY/*أتمنى, *etc.)* with their derivations. In (F126–F127). Furthermore, we checked for the existence of inappropriate words wrongly used to start or conclude an Arabic essay such as *(bsm Allh AlrHmn AlrHym/* بسم الله الرحمن الرحيم, *OmA bEd/* أمابعد, *AlHmdllh rb AlEAlmyn/* الحمدلله رب العالمين, *SlY Allh wbArk/*صلى الله وبارك, *etc.)* which usually used in other types of writing in Arabic.

Arabic Lexicon Features (F128–F133): we relied on four Arabic lexicons to extract (F128–F133): *The Contemporary Arabic Language Dictionary*; alWaseet lexicon; the Arabic Wordlist for spellchecking (Attia, Pecina, & Samih, 2012) which contains 9 million words automatically generated from the AraComLex open-source finite-state transducer (30,000 lemmas), and a billion-word corpus; and Obsolete Arabic Words (Attia, Pecina, Toral, Tounsi, & Genabith, 2011) which includes obsolete words or words that are not in contemporary use, in the Buckwalter Morphological Analyzer database. As separate features, we checked the number of words that belong to each of the obsolete list, alWaseet, and Contemporary lexicons, as well as words that do not belong to the spellchecking list. We proposed these features to evaluate spelling and style criteria.

Punctuation features (F134–F161): for each punctuation mark, we counted its frequency in the essay. So, in separate features, we counted the frequency of each of the following: question mark (F134), exclamation (F135), period (F136), comma (F137), semicolon (F138), quotation mark (F139), parentheses (F140), dash (F141) and colon (F142). Also, we counted the number of times the writer repeats the same punctuation mark in one use (F142) such as a repeating period (....) or question mark (???), which represents one of the common mistakes in Arabic writing. Furthermore, we have additional features related to each punctuation mark that can be broken down into three categories: correct use, missing use, and incorrect use of a punctuation mark. For the correct use of a question mark (F143), we counted the number of times a sentence contains question tools, including *hl/*هل, *kyf/*كيف, *mA*A/*ماذا, *lmA*A/*لماذا, *mA/*لم, *km/* كم, *mtY/*متى, or *Ayn/*أين, along with a question mark. The question mark was considered missing if a sentence contained one of the question tools yet was missing a question mark (F144). In case of an incorrect usage, we counted the number of times a sentence contained a question mark without the existence of a question tool (F145).

For correct use of the exclamation mark, we counted the number of times an exclamation existed in a sentence containing one of the exaggerating styles, such as *yAlyt/*ياليت, *b}s/*بئس, *rA}E/*رائع, or *llh dr~/*لله در, or contained a word in the pattern *mA OfEl/*ما أفعل (F146). A missing use was considered if one of these keywords existed while the exclamation mark was absent (F147). For an incorrect use, we counted the number of times an exclamation existed while the previous indicators were missing (F148). For semicolons, to detect correct usage we checked the word following the semicolon. If the word had a causative meaning, such as *lOn/*لأن, *bsbb/*بسبب, *ky/*لكي, or the word started with the clitic *l/*لـ or *f/*ف, then the number of correct uses of the semicolon increases (F149). A missing use considered when the previous indicators existed and the semicolon mark was missing (F150). For the wrong use of the semicolon, we counted the number of times a semicolon was not followed by those causative indicators (F151). Also, we considered comma as an incorrect use if it was involved in the paragraph containing discourse connectives as presented in Alsaif (2012) (F152). A comma also was classified as misused if a paragraph containing connectives was missing a comma (F153), and an incorrect use was considered in the case of a comma followed by causative indicators.

For the period mark, we counted the number of correct usages if each paragraph ended with a period (F154), and we increased the number of missing period marks if the paragraph did not end with a period. An incorrect usage of the period was

considered when a paragraph contained a period before the end (F155). Moreover, we counted the number of correct uses of the colon by checking the existence of some words such as *mvAl/*مثال, *Al|typ/*التالية, *AltAlyp/*الآتية, or *mAyly/*مايلي with a colon mark. Also, a colon was considered correct if it was involved in a sentence containing a word referring to reported speech based on list of attribution cues, as seen in a study by (Alsaif et al. 2018) (F156). We counted the number of times a colon was missing when indicators were present yet a colon was absent (F157). Conversely, the use of a colon was considered wrong if a colon mark was present while indicators were missing (F158). Finally, for quotation marks, we counted the number of correct uses by checking cases in which quotation marks were preceded by one of the words referring to reported speech, using the list of attribution cues and where opening and closing pairs of quotations were placed (F159). A missing use of a quotation mark was considered when at least one word of the list mentioned was present yet quotation mark pairs or a single one was missing (F160). An incorrect use was considered when a quotation mark was present without a cue (F161).

D. Semantic features

Unlike traditional dictionaries, WordNet is organized by semantic relations between synsets. In this work, we use the Arabic WordNet AWN (*Arabic WordNet—Global WordNet Association*, 2013) alongside the NLTK module to extract our semantic features based on some relations such as synonyms and antonyms between essay sentences. First, we counted number of matched words not only between two adjacent sentences but also all essay sentences even matched words within the same sentence (F162) to evaluate the coherence of the essay. However, as a sentence can be expressed using different synonyms, we counted the number of synonyms in the entire essay (F203) and between all sentences using Arabic WordNet (AWN). These features also were applied over essay paragraphs to measure the similarities between essay parts (F163–F168). Moreover, we used one of AraVec models (Bakr, Mohammad, Eissa, & El-Beltagy, 2017), that proposed Arabic Word Embedding for use in Arabic NLP. For each sentence, we counted the similarity between its words and all other sentences words (F169).

E. Discourse features

As some criteria require examination between essay parts, such as coherence criterion, we propose the following features:

Arabic discourse connectives (F170–F188): we counted the number of connectives (F170) according to the list of unambiguous discourse connectives (Alsaif, 2012) in terms of discourse function, so that at least 90% of their occurrences in the Leeds Arabic Discourse Treebank (LADTB), were annotated as discourse connectives. Furthermore, we counted number of unique connectives in the overall essay (F171). We also distributed these two features over essay paragraphs and sentences (F172–F179). In addition, for each paragraph, we counted the ratio of connectives (F180) and unique connectives to the paragraph's words (F181). We then counted the ratio of number of words located between two discourse connectives to the number of connectives per paragraph in the essay (F182). In the same way, we counted the ratio of connectives (F183) and unique connectives to the sentences' words (F184). Moreover, we counted the number of times punctuation was not followed by a connective or conjunction (F202). Using the POS tag provided by MADAMIRA, we checked the number of words with conjunction tags in the overall essay (F185) as well as the number of unique conjunctions (F186). Likewise, we applied the same connectives features but with conjunctions tools in (F187) and (F188).

5 Experiments and Results

All experiments in this study were carried out using the WEKA tool (Witten et al., 1999), based on a tenfold cross-validation for the entire dataset. We built a specific model for each criterion: spelling, coherence, structure, punctuation marks, and style. Then, we computed the models' results to predict the overall score. During each model development, we considered the size of our dataset, the appropriate features, and the number of features to achieve the most accurate model with the minimum number of features possible to avoid overfitting problems (Ying, 2019). It is worth noting that we only included the effective features in each model rather than including all features.

To evaluate the system's performance, we used accuracy "Acc," which refers to the number of essays correctly evaluated by the system, as well as

Pearson's correlation *r* to measure the relationship and association between manual scores and system scores (Benesty, Chen, Huang, & Cohen, 2009). As our dataset contains fractions, and to align with scoring in similar studies (Alghamdi et al., 2014; Azmi, Al-Jouie, & Hussain, 2019; Alqahtani & Alsaif, 2019), we considered a threshold value ***t*** in our results. Therefore, an essay is considered as correctly evaluated if the difference between the manual score and the system score does not exceed ***t***. Alghamdi et al., (2014) set ***t*** to be approximately 17% of the overall score, whereas Azmi, Al-Jouie, and Hussain (2019) set ***t*** to be 25% of the essay score. In our case, although our dataset contains many fractional numbers, we will show the results when ***t*** = 17% and ***t*** = 25% in specific criteria scores and the overall score. Table 1 shows score distributions in our dataset and the threshold values.

Criteria	score	Threshold at *t*=17%	Threshold at *t*=25%
spelling	4	0.68	1
structure	4	0.68	1
coherence	4	0.68	1
Punctuation marks	2	0.34	0.5
style	2	0.34	0.5
Overall score	16	2.72	4

Table 1. Score distributions and the threshold value per criterion.

A. Spelling model

We attempted many different features in the spelling model. However, the effective features were found at the surface level, lexical level, syntactical, and morphological levels as follows: (F1–F23), F29, F100, F101, F114, F117, F123, F133, F200, and F201, while the significant features were based on features related to the FARASA spellchecker (F102–F105). Using this model, the number of essays that were evaluated correctly in spelling represent 58% of our dataset when ***t*** = 17%, while it increased to 77% when ***t*** = 25%, as shown in Table 2. The variant between these two results returns to the value of the threshold and the number of fractions scores in spelling in our dataset. Further, the model achieved 0.65 when ***t*** = 17% and 0.72 when ***t*** = 25% in correlation *r* to manual evaluation. However, since the model is almost based on FARASA features, we analyzed this tool over our dataset by detecting how many times FARASA detects actual mistakes and corrects them in the right way; how many times FARASA detects actual mistakes but does not correct them as what should be; and how many times FARASA detects a correct word as a mistake, which were 2130, 85, and 250 cases, respectively. However, there were 120 words corrected by FARASA only because missing spaces in some cases that usually difficult to detect by humans such in *mAh*A/what* /ماهذا.

B. Structure model

Since a well-structured essay should contain four parts; title, introduction, body and conclusion, we included surface features (F3) and (F16) that refer to the number of paragraphs and check if the first paragraph is less than or equal to 10 words. Also, we include lexical features (F124 and F126) which related to check the existence of some keywords usually used in the introduction and conclusion parts. This model achieved 78% in Acc and 0.74 in correlation *r* when ***t***=17% and 91% in *Acc* and 0.86 in correlation *r* when ***t***=25 %. The significant feature was (F16) which refers to check if the first paragraph less than or equal 10 words which assists to indicate the title of the essay.

C. Coherence model

Coherence criterion used to evaluate the extent to which essay parts is related to the title, cohesion between essay parts, using the appropriate discourse connectives and diversity in connectives. Therefore, we included surface features (F3, F19, F20, F6–F10, F15, F12, F13), lexical features (F124 and F125) and syntactic features (F23–F99). Most of these features are generic and hold information about essay parts (lengths, general syntactic characteristics). These features are utilized to figure out essay structure which in turn assists to predict the extent of the appropriate coherence. Furthermore, we include discourse features (F170, F171, F202, F179, F137, F177,

Criteria	Level of features included per criterion	*t* = 17%		*t* = 25%	
		Acc	*r*	*Acc*	*r*
Spelling	Surface+Lex+(Syn and Morph+Spelling)	58%	0.65	77%	0.72
Structure	Surface +Lex	78%	0.74	91%	0.86
Coherence	Surface+Lex+Syn and Morph+Sem+Disc	79%	0.65	87%	0.69
Punctuation marks	Surface+(Lex/punctuations)+Syn/pos+Disc + Sem	75%	0.53	93%	0.74
Style	Surface+Lex+Syn+Disc+Sem	65%	0.57	78%	0.65
Overall score	A combination of the essay evaluation results in the previous criteria	90%	0.82	96%	0.87

Table 2. Features levels used for criteria modeling with their results and the overall score considering the threshold.

F187, F180–F182) since they related to connectives between essays parts in addition to the semantic features presented in (F62, F163, F169, F168 and F203) to prevent relying on only the matched words.

This model achieved 79% in accuracy and 0.65 in *r* correlation in case of ***t*** = 17% while it increased to 87% and 0.69 in accuracy and correlation *r* respectively when ***t*** = 25%. However, in some cases detecting cohesion automatically is very difficult especially if the unrelated idea is expressed within a short sentence.

D. Style model

As essay with a good style does not include repeated words without using synonyms and does not contain informal words while there is a good choice of words and diversity in the length of sentences (Ibrahim, 2006). We included surface features (F1, F5, F4, F21, F10), which predominantly investigate the length of paragraphs and sentences, lexical features from (F123, F128–F133, F134–F142) that check punctuation use, and the number of words that may affect the style. We also check morphological features (F117) as they also affect the form of words, which in turn sometimes leads to unknown or informal words. Additionally, we include discourse features and connectives (F170–F180) since punctuation might be omitted by a writer, hence there are no indications of paragraph and sentence lengths. Also, we add semantic features (F162–F168 and F203) to investigate synonymous words. However, discourse and semantics have the most impact in score prediction where the significant features were the number of discourse connectives (F170) and the number of synonyms in the whole essay (F203). The style achieved 65% in *Acc* and 0.57in correlation when ***t***=17% while it increased to 78% in accuracy and 0.65 in correlation when ***t***=25% as shown in Table2.

E. Punctuation marks model

We included surface features (F6–F9 and F12), general syntactic features such as POS (F23–F29, F31–F58) and features related to punctuation marks which refer to correct, wrong, and missing use for all marks (F134–F161). Also, we included discourse connectives (F202, F184, F187), as they separate the essay to sentences/clauses and that may assist to detect some types of mistakes such as the omission of placing comma. As each punctuation mark has certain purposes (e.g., a period used at the end of a sentence, or a comma used within a sentence to separate it into clauses), we tried to figure out the semantic impact by including features (F162–F168) which refer to the similarity between sentences. We noted that the most two significant features were (F202) that refers to the number of punctuations not followed by conjunction or discourse connectives and (F184), which refers to the ratio of unique connectives to paragraph length. The punctuation model achieved different results due to the threshold value, as in Table 2.

In the overall score, we combined the results obtained by all the previous criteria as predicted by the models, then we calculated accuracy which was 90% and 96% when ***t*** = 17% and ***t*** = 25% respectively.

In comparison to the similar studies (Alghamdi et al., 2014) and (Azmi, Al-Jouie, & Hussain, 2019), our system utilizes a wide range of features to evaluate the common criteria and can

provide an evaluation of a specific criterion further to the overall score. It also does not need to train on representative-domain essays. Unfortunately, we cannot provide an accurate comparison because their datasets cannot be accessed. In addition, our system applied to a larger dataset than that used in Alqahtani and Alsaif (2019) and it supports semantic aspects which not covered by their system.

6 Conclusion

This paper introduced an Arabic essay evaluation system based on the SVR algorithm and features from different linguistic levels. Separately, we conducted experiments to predict five criteria scores; spelling, structure, coherence, style, and punctuation marks. The essay holistic score was assigned by a combination of the previous criteria scores. The experiments conducted on our dataset consisted of 200 essays. In the overall evaluation, the proposed system achieved 96% accuracy and 0.87 in correlation with manual evaluation, while it achieved 77%, 91%, 87%, 78%, and 93% in accuracy for spelling, structure, coherence, style, and punctuation marks, respectively. In the future, we look forward to expanding our dataset as our system performance improved by increasing the dataset size from 100 essays to 200 essays. Furthermore, we intend to involve some criteria that have been annotated by humans but not yet automated, such as grammar. Similar studies in foreign languages have had promising results by applying deep learning algorithms while it is unexplored in AES for Arabic writing. Therefore, we believe it is worth to apply deep learning algorithms utilizing the features extracted in this work.

References

Al-Shalabi, E. F. (2016). An automated system for essay scoring of online exams in Arabic based on stemming techniques and Levenshtein edit operations. *International Journal of Computer Science Issues*, *13*(5), 45–50.

Alghamdi, M., Alkanhal, M., Al-Badrashiny, M., Al-Qabbany, A., Areshey, A., & Alharbi, A. (2014). A hybrid automatic scoring system for Arabic essays. *AI Communications*, *27*(2), 103–111. https://doi.org/10.3233/AIC-130586

Ali, K. (2018). Inna and Its Sisters. Retrieved from https://arabicblog.info/inna-and-its-sisters

Ali, K. (2018). Kana and Its Sisters. Retrieved from https://arabicblog.info/kana-and-its-sisters

Ali, K. (2019). Fi'l Mudhari. Retrieved from https://arabicblog.info/fil-mudhari

Alikaniotis, D., Yannakoudakis, H., & Rei, M. (2016). Automatic Text Scoring Using Neural Networks. https://doi.org/10.18653/v1/P16-1068

Alqahtani, A., & Alsaif, A. (2019). Automatic Evaluation for Arabic Essays: A Rule-Based System. In *2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)* (pp. 1–7). https://doi.org/10.1109/ISSPIT47144.2019.9001802

Arabic WordNet—Global WordNet Association. (2013). Retrieved from http://globalwordnet.org/resources/arabic-wordnet/

Alsaif, A. (2012). Human and Automatic Annotation of Discourse Relations for Arabic (Doctoral dissertation, University of Leeds, Leeds, England). Retrieved from http://etheses.whiterose.ac.uk/3129/

Al-Saif, A., Alyahya, T., Alotaibi, M., Almuzaini, H., & Algahtani, A. (2018). Annotating Attribution Relations in Arabic. *LREC*.

Attia, M., Pecina, P., & ASamih, Y. (2012). Improved Spelling Error Detection and Correction for Arabic, *4*(December), 103–112.

Attia, M., Pecina, P., Toral, A., Tounsi, L., & Genabith, J. Van. (2011). A Lexical Database for Modern Standard Arabic Interoperable with a Finite State Morphological Transducer, 98–118.

Awad, M., & Khanna, R. (2015). Support Vector Regression. In *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers* (pp. 67–80). Berkeley, CA: Apress. https://doi.org/10.1007/978-1-4302-5990-9_4

Azmi, A. M., Al-Jouie, M. F., & Hussain, M. (2019). AAEE - Automated evaluation of students' essays in Arabic language (July). https://doi.org/10.1016/j.ipm.2019.05.008

Bakr, A., Mohammad, S., Eissa, K., & El-Beltagy, S. R. (2017). AraVec : A set of Arabic Word Embedding Models for use in Arabic NLP ScienceDirect (November). https://doi.org/10.1016/j.procs.2017.10.117

Benesty, J., Chen, J., Huang, Y., & Cohen, I. (2009). Pearson Correlation Coefficient. In *Noise Reduction in Speech Processing* (pp. 1–4). Springer: Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-00296-0_5

Buckwalter, T. (2004). Buckwalter Arabic morphological analyzer version 2.0. Linguistic Data

Consortium, University of Pennsylvania, 2002. LDC catalog no.: LDC2004l02. Technical report, ISBN 1-58563-324-0.

Burstein, J. (2003). The e-rater scoring engine: Automated essay scoring with natural language processing. *Automated Essay Scoring: A Cross-Disciplinary Perspective*, 113–121.

Chung, G., & O'Neil, G. (1997). Methodological Approaches to Online Scoring of Essays, CSE Technical Report 461, University of Southern California CRESST, December, *Center for the Study of Evaluation, CRESST*, *1522*(310).

Dikli, S. (2006). An Overview of Automated Scoring of Essays. *Journal Of Technology Learning And Assessment*, *5*(1), 2006–12. Retrieved from http://www.jtla.org

Dong, F., & Zhang, Y. (2016). Automatic Features for Essay Scoring—An Empirical Study. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, *1966*, 1072–1077. https://doi.org/10.18653/v1/D16-1115

Elliot, S. (2003). IntelliMetric: From here to validity. in M. Shermis and J. Burstein (eds.) *Automated Essay Scoring: A Cross-Disciplinary Perspective*, Routledge, 71–86. DOI: https://doi.org/10.4324/9781410606860

FARASA: Advanced Tools for Arabic. (2019). Retrieved from http://qatsdemo.cloudapp.net/farasa/

Gomaa, W. H., & Fahmy, A. A. (2014). Automatic scoring for answers to Arabic test questions. *Computer Speech and Language*, *28*(4), 833–857. https://doi.org/10.1016/j.csl.2013.10.005

Ibrahim, M. (2006). Looks at the technical article in modern Arabic literature (نظرات في المقال الفني في الأدب العربي الحديث). pp.21–22.

Janda, H. K., Pawar, A., Du, S., & Mago, V. (2019). Syntactic, Semantic and Sentiment Analysis: The Joint Effect on Automated Essay Evaluation. *IEEE Access*, *7*, 108486–108503.

Kukich, K. (2000). Beyond automated essay Scoring. IEEE Intelligent Systems and Their Applications. https://doi.org/10.1109/5254.889104

Lemaire, B., & Dessus, P. (2001). A System to Assess the Semantic Content of Student Essays. *Journal of Educational Computing Research*, *24*(3), 305–320. https://doi.org/10.2190/G649-0R9C-C021-P6X3

Nahar, I. K. M., & Alsmadi, I. M. (2009). The Automatic Grading for Online exams in Arabic with Essay Questions Using Statistical and computational linguistics Techniques, *1*(2), 215–220.

Pasha, A., Al-Badrashiny, M., Diab, M., Kholy, A. El, Eskander, R., Habash, N., … Roth, R. M. (2014). MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic. *Proceedings of the 9th Language Resources and Evaluation Conference (LREC'14)*, 1094–1101.

SAFAR: Software Architecture For ARabic. (2013). Retrieved from http://arabic.emi.ac.ma/safar/?q=examples#

Shehab, A., Faroun, M., & Rashad, M. (2018). An Automatic Arabic Essay Grading System based on Text Similarity Algorithms, (April). https://doi.org/10.14569/IJACSA.2018.090337.

Surya, D., Madala, V., Krishna, S., Surya, D., Madala, V., Gangal, A., … Sureka, A. (2018). An empirical analysis of machine learning models for automated essay grading.

Witten, I. H., E. Frank, L. E. Trigg, M. A. Hall, G. Holmes and S. J. Cunningham. (1999). Weka: Practical machine learning tools and techniques with Java implementations. Proc ICONIP/ ANZIIS/ANNES99 Future Directions for Intelligent Systems and Information Sciences, pp. 192–196.

Ying, X. (2019). An Overview of Overfitting and its Solutions: IOP Conf. Series: Journal of Physics: Conf. Series 1168. 2019.

Semantic Extractor-Paraphraser based Abstractive Summarization

Anubhav Jangra*
IIT Patna, India
anubhav0603@gmail.com

Raghav Jain*
DTU, India
raghavjain106@gmail.com

Vaibhav Mavi*
IIT Delhi, India
vaibhavg152@gmail.com

Sriparna Saha
IIT Patna, India
sriparna.saha@gmail.com

Pushpak Bhattacharyya
IIT Bombay, India
pushpakbh@gmail.com

Abstract

The anthology of spoken languages today is inundated with textual information, necessitating the development of automatic summarization models. In this manuscript, we propose an extractor-paraphraser based abstractive summarization system that exploits semantic overlap as opposed to its predecessors that focus more on syntactic information overlap. Our model outperforms the state-of-the-art baselines in terms of ROUGE, METEOR and word mover similarity (WMS), establishing the superiority of the proposed system via extensive ablation experiments. We have also challenged the summarization capabilities of the state of the art Pointer Generator Network (PGN), and through thorough experimentation, shown that PGN is more of a paraphraser, contrary to the prevailing notion of a summarizer; illustrating it's incapability to accumulate information across multiple sentences.

1 Introduction

Over the past few years, the Internet has become the most convenient and preferred form of information sharing worldwide. The evolution of technology has made it possible for anyone to convey their knowledge, opinions and ideals to the world, resulting in an increasing surge of information hindering users from accessing desired content. This increasing need to obtain key information makes the task of summarization paramount. Text is the most widely adopted form of communication, be it for personal messaging[1] or for broadcasting, owing to its ability to convey almost any concept, its general flexibility to suit everyone's needs, and its less storage requirement (opposed to other modes of communication like audio and video). Text summarization is a problem at the very core of natural language processing, and has various applications in the spoken languages, including summarization of conversations, and public speeches.

Some works have been done in the field of reinforcement learning based text summarization (Dong et al., 2018; Liu et al., 2018), the most prominent architecture being extractor-abstractor (EXT-ABS) model (Chen and Bansal, 2018). Inspired from this architecture, in this manuscript, we have proposed an extractor-paraphraser system that uses semantic information overlap as the underlying guidance strategy. The model is further enhanced to surpass its limits using reinforcement learning, for which we have proposed a novel semantic overlap based reward function. Word Mover Similarity (WMS) (Clark et al., 2019) is utilized to evaluate semantic similarity across generated sentences and the true ground truth summary sentences.

We assume that paraphrasing is a relatively simpler task than abstractive summarization, with the underlying intuition that paraphrasing is a subproblem within abstractive summarization. To bolster our hypothesis, experiments are conducted on the extractor-abstractor (EXT-ABS) model (Chen and Bansal, 2018) and the Pointer Generator Network (PGN) (See et al., 2017), which is used as the basic abstraction unit in the former architecture. The results are rather staggering and reveal that the PGN model also paraphrases input document sentences, albeit implicitly. The major contributions of the paper are as follows:

- A novel semantic overlap based reward function is proposed for reinforcement of extractor-paraphraser model.
- To the best of our knowledge, we are the first ever to discover the fact that PGN networks are indeed doing an implicit extraction-paraphrasing operation, revealing the true nature of existing abstractive summarization models.

* means equal contribution.

[1] https://news.gallup.com/poll/179288/new-era-communication-americans.aspx

Proceedings of the 17th International Conference on Natural Language Processing, pages 191–199
Patna, India, December 18 - 21, 2020.

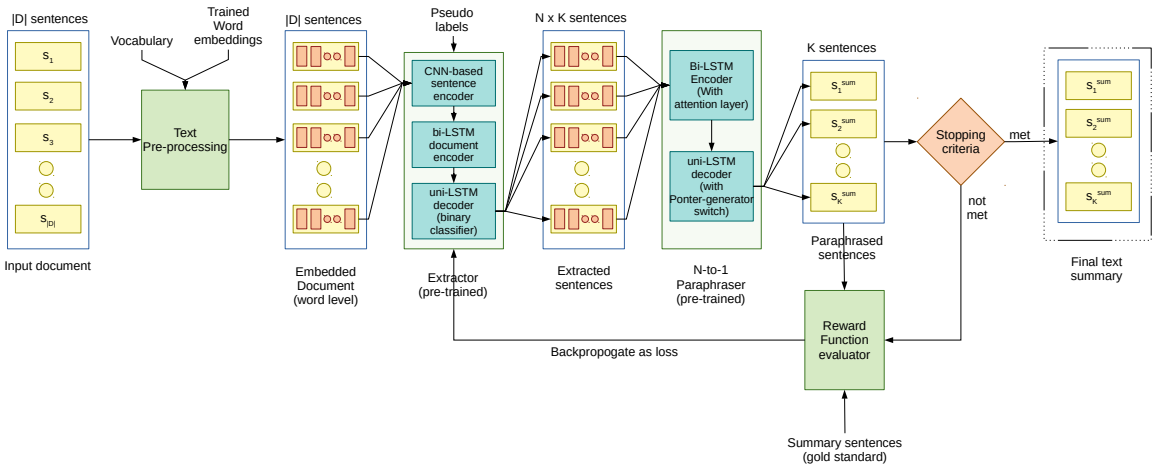


Figure 1: Proposed model architecture.

The rest of this paper is structured as follows: In Section 2 we have discussed related works of automatic text summarization. In Section 3 we have described the proposed model, and in Section 4 we have stated the experimental setup and the datasets used. A thorough discussion and state the results are provided in Section 5, followed by the conclusion and future work in Section 6.

2 Related Work

Automatic text summarization has been extensively researched over more than three decades, and has shown a lot of progress and promise over the course of time. Various approaches have been explored to tackle both extractive and abstractive summarization. Initial research (Paice, 1990; Kupiec et al., 1995) focused on extractive summarization due to its easier setup. Various techniques ranging from integer linear programming (Galanis et al., 2012), graph based approaches (Mihalcea and Tarau, 2004; Mihalcea, 2004), genetic algorithms (Saini et al., 2019a,b), and neural networks (Nallapati et al., 2017; Zhang et al., 2016) have been adopted to solve the extractive summarization task. The majority of the research in abstractive summarization revolves around deep learning (See et al., 2017; Chopra et al., 2016; Nallapati et al., 2016). Liu et al. (2018) proposed a generative adversarial network based model to generate document abstracts. A handful of works however also use ILP (Banerjee et al., 2015) and graph-based (Ganesan et al., 2010) techniques to attempt to solve the problem. A lot of domain specific summarization techniques have also been explored, like radiology findings summarization (Zhang et al., 2018), across-time summarization (Duan and Jatowt, 2019), movie review summarization (Zhuang et al., 2006), book summarization (Mihalcea and Ceylan, 2007), and customer review based opinion summarization (Pecar, 2018). Lately, multi-modal summarization (Jangra et al., 2020a,b; Zhu et al., 2020; Saini et al., 2020) has also gained popularity .

Recently, people have also explored reinforcement learning to tackle the problem of automatic text summarization in both extractive (Dong et al., 2018; Gao et al., 2019) and abstractive domains (Xiao et al., 2020; Chen and Bansal, 2018). Chen and Bansal (2018) have proposed an extractor-abstractor architecture, separating the relevant data searching part and the paraphrasing part to individual modules. In this work, we have proposed a system inspired from Chen and Bansal (2018), stressing on the significance of semantic information over the traditional syntactic overlap. The literature on text summarization is rich, and has an abundance of survey papers (Yao et al., 2017; Gambhir and Gupta, 2017) to get an in depth overview of the domain.

3 Proposed Method

Problem Definition: Given the training data {X, Y} where $X = \{d_1, d_2, ..., d_N\}$ is the set of input documents and $Y = \{y_1, y_2, ..., y_N\}$ is the set of corresponding output summaries, the task of automatic summarization is defined as the problem of discovering a function $f : X \mapsto Y$, such that $f(d_i) = y_i; \forall i \in \{1, 2, ..., N\}$.

We have proposed an extractor-paraphraser framework, which is inspired from the extractor-abstractor (EXT-ABS) framework introduced by

Chen and Bansal (2018). The summarization function $f(.)$ is approximated as the composition, $f(d_i) = h(g(d_i))$, where the functions $g(.)$ and $h(.)$ are modeled as the *extractor* and the *paraphraser* components of the model, respectively. Given an input document $d_i = \{s_1^{d_i}, s_2^{d_i}, ..., s_{|d_i|}^{d_i}\}$, the extractor $g(.)$ extracts relevant sentences, acting as the primary noise filter. These extracted set of sentences are fed to the paraphraser $h(.)$, which accumulates the information into a concise gist of the extracted sentences, simulating the natural language generation module in the proposed system (Fig. 1). The paraphraser in our system is capable of summarizing multiple extracted sentences to generate one sentence, in contrast with its predecessor 'abstractor' from EXT-ABS model (Chen and Bansal, 2018) which rephrases one extracted sentence at a time. The proposed framework consists of an 'n-to-one paraphraser', that compiles n sentences into one sentence, generating a richer summary. Formally, given a document $d_i = \{s_1^{d_i}, s_2^{d_i}, ..., s_{|d_i|}^{d_i}\}$, the extractor is defined as:

$$\begin{aligned} &g : X \mapsto Z^{|y_i| \times n}, \\ &\text{s.t. } g(d_i) = \{k_{j,l} \,|\, 1 \leq k_{j,l} \leq |d_i|\}_{l=1,j=1}^{n,|y_i|} \end{aligned} \tag{1}$$

where $k_{p,q}$ represents the index of the q^{th} extracted sentence corresponding to the p^{th} sentence in the final gold summary. For modelling the same, we have used an encoder-decoder model, where the encoder consists of a temporal convolutional model (Kim, 2014) cascaded with a bidirectional LSTM network (Schuster and Paliwal, 1997) and the decoder is a uni-directional LSTM model (Hochreiter and Schmidhuber, 1997) (Fig. 1).

The paraphraser function $h(.)$ is defined as follows:

$$h(\{k_{j,l}\}_{l=1,j=1}^{n,|y_i|}, d_i) = \{p(\oplus_{l=1}^{n} s_{k_{j,l}}^{d_i})\}_{j=1}^{|y_i|} \tag{2}$$

where $\oplus$ represents the concatenation of sentences, $k_{j,l}$ represents the extractor output and p is written as:

$$p(s_j^{ext}) = s_j^{y_i} \tag{3}$$

where $s_j^{ext} = \oplus_{l=1}^{n} s_{k_{j,l}}^{d_i}$ and $s_j^{y_i}$ is the j^{th} gold summary sentence. As shown in Fig. 1, a pointer-generator framework (See et al., 2017) is used to model the paraphraser.

3.1 Training the Submodules

In order to train the paraphraser beforehand, we need to mimic the output of an extractor[2]. To fulfil this requirement, *exemplary-extracted sentences* are generated using gold summaries. Word-mover distance (WMD) (Kusner et al., 2015) is adopted to create these exemplary-extracted sentences, with the motivation that it captures the semantic overlap between sentences better than its predecessors. For a training pair $\{d, y\}$, the labels are generated as:

$$\begin{aligned} &\forall l \in \{1, 2, ..., n\} : \\ &\quad \forall s_j \in y : \\ &\quad\quad k_{j,l} = argmin_i\{WMD(s_i^d, s_j^y)\}; \\ &\quad\quad \forall s_i^d \in d - \{s_{k_{j',0}}^d\}_{j'=1}^{j-1} \end{aligned} \tag{4}$$

where $k_{j,l}$ represents the l^{th} exemplary- extracted sentence corresponding to the j^{th} gold summary sentence, s_j^y.

The n exemplary-extracted sentences corresponding to each summary sentence are concatenated and fed into the paraphraser as the input and the summary sentence is fed as the output. We argue that this should allow the paraphraser to generate information rich summaries. To facilitate the expected behaviour during the testing phase, we require the extractor to feed the paraphraser with input similar to the exemplary-extracted sentences. Hence, we first pre-train the extractor on these exemplary-extracted sentences as opposed to random initialization. Cross-entropy loss is used for training the paraphraser and pre-training the extractor.

3.2 Extractor agent

Enhancement using reinforcement learning: Here, the extractor is trained as part of an actor-critic model (Mnih et al., 2016) which takes an action based on the current state and current value of parameters to maximize a given reward at each time step, where the action is to extract n sentences, $\{k_{t,l}\}_{l=1}^{n}$, and the state refers to the set of document sentences, d_i, and already extracted sentences, $\{d_{k_{1,l}}, d_{k_{2,l}}, ..., d_{k_{t-1,l}}\}$; $l \in \{1, 2, ..., n\}$. The predicted sentences are concatenated and passed to

[2]Note that the extractor's output can also be used to train the paraphraser at this step, however, since the extractor's weights get fine-tuned using reinforcement learning at a later stage, training paraphraser on extractor's non-ideal outputs might lead to unsatisfactory performance of the paraphraser.

the paraphraser to get an output sentence. A novel semantic-based reward function using word mover distance (WMD) is used as the reward function. Since the reward is to be maximised, WMD needs to be converted to a similarity function, for which, a generalised version of word mover similarity (WMS), proposed in (Clark et al., 2019), is used. Formally, at a time step t, given an action $j_{t,l}$, and a summary sentence, s_t, the short term reward, r is calculated as[3]:

$$r = \frac{a+1}{a + e^{b \times WMD(s_t^y, p(s_t^{ext}))}} \quad (5)$$

where $s_t^{ext} = \oplus_{l=1}^{n} s_{k_{t,l}}^{d}$, and $\oplus$ represents the concatenation of sentences, $\{s^d, s^y\}$ are the sentences belonging to a training pair and a and b are the hyper-parameters introduced[4].

To avoid redundant phrases and words, tri-gram avoidance through beam search (Paulus et al., 2017) is applied at a sentence level. For a fair comparison, details regarding the beam search reranking and other nuances of implementation are kept the same as (Chen and Bansal, 2018).

4 Experiments

4.1 Dataset

For all the following experiments we have used the CNN / DailyMail dataset (Nallapati et al., 2016), which contains online news articles, with the bullet highlights treated as the gold standard summaries. The experiments in this work are conducted on the non-anonymized version of this dataset. The dataset consists of 277,226 training, 13,368 validation and 11,490 test article-summary pairs. An article contains $\sim$780 tokens per document, whereas the summary consists of $\sim$ 56 tokens with the average number of sentences per summary being $\sim$3.75. An article sentence, on average contains $\sim$30 tokens.

4.2 Comparative methods

To highlight the superiority of the proposed semantic-overlap based methodology over existing syntactic measures, we set the value of $n = 1$ for initial experiments. To compare the complexity of the summarization task with paraphrasing objectively, the experiments are further extended for $n = 2$ as well [5]. We evaluate our model with sufficient baselines [6], including:

Pointer-Generator network [PGN]: An encoder-decoder attention based framework for abstractive summarization (See et al., 2017).

Pre-trained extractor - Extractor-Abstractor framework [EXT − ABS (Ext only)]: The pre-trained extractor of the extractor-abstractor framework proposed by Chen and Bansal (2018) on ROUGE-L based exemplary-extracted sentences.

Extractor-Abstractor framework without reinforcement [EXT − ABS (w/o RL)]: The extractor-abstractor framework proposed by Chen and Bansal (2018) at a stage before reinforcing the extractor.

Extractor-Abstractor framework [EXT− ABS + $\mathbf{RL_X}$]: The complete extractor-abstractor framework proposed Chen and Bansal (2018) including X as the reward function for reinforcement learning (X is either ROUGE-L or the reward function defined in Eq. 5) along with beam search.

Pre-trained extractor - One-to-one extractor-paraphraser model [O2O (Ext only)]: The pre-trained extractor of the proposed extractor-paraphraser framework (with $n = 1$) on WMD based (Eq. 4) exemplary-extracted sentences.

One-to-one extractor-paraphraser model without reinforcement [O2O (w/o RL)]: A particular setting of the proposed methodology where $n = 1$, without any reinforcement or beam search.

One-to-one extractor-paraphraser model [O2O+ $\mathbf{RL_X}$]: A particular setting of the proposed methodology where $n = 1$, including extractor, paraphraser, and X as the reward function for reinforcement learning (X is either ROUGE-L or the reward function defined in Eq. 5) and beam search.

Pre-trained extractor - Two-to-one extractor-paraphraser model [$\mathbf{M2O_{n=2}}$ (Ext only)]: The pre-trained extractor[7] of the proposed extractor-paraphraser framework (with $n = 2$) on WMS based (Eq. 4) exemplary-extracted sentences.

[3]Note that we obtain the word mover similarity proposed in (Clark et al., 2019) for the case $a = 0$, and $b = 1$.

[4]The hyperparameters are set to $a = 1$ and $b = 0.5$ after extensive experimentations, and are used throughout this paper.

[5]We have limited our work to $n = 2$ since the Two-to-one baselines did not perform efficaciously. Experiments for $n > 2$ would be done in future works.

[6]Statistical analysis on all the variations of the proposed model has been done.

[7]For fair comparison of extraction capabilities across all models, we limit the model to output 4 sentences during evaluation on test dataset.

Table 1: Evaluation scores for the generated text summary using ROUGE, METEOR and Word Mover Similarity (WMS). 'Id-ext' refers to the ideal extractor experiments. The '-' denotes unavailability of a score.

Models	ROUGE-1	ROUGE-2	ROUGE-L	METEOR	WMS
Extractive baselines					
$EXT-ABS\,(Ext\,only)$	40.17	18.11	36.41	**22.81**	-
$O2O\,(Ext\,only)$	**40.97**	**18.42**	**37.35**	21.86	**14.28**
$M2O_{n=2}\,(Ext\,only)$	40.19	17.98	36.60	21.62	14.24
Abstractive baselines					
PGN	39.53	**17.28**	36.38	18.72	13.36
$EXT-ABS\,(w/o\,RL)$	38.38	16.12	36.04	**19.39**	-
$O2O\,(w/o\,RL)$	**39.82**	17.05	**37.21**	19.24	**13.81**
$M2O_{n=2}\,(w/o\,RL)$	32.82	11.29	31.08	16.13	12.92
Reinforced models					
$EXT-ABS+RL_{ROUGE}$	40.88	17.8	38.53	20.38	13.7
$O2O+RL_{ROUGE}$	**41.32**	**18.16**	**38.89**	20.52	14.56
$EXT-ABS+RL_{WMS}$	40.82	17.82	38.45	**21.47**	14.42
$O2O+RL_{WMS}$	41.2	18.12	38.81	21.34	**14.6**
$M2O_{n=2}+RL_{ROUGE}$	39.71	16.7	37.32	18.25	13.52
Ablation experiments (Ideal Extractor)					
$Id-ext\,EXT-ABS$	49.73	**26.53**	47.2	24.36	19.34
$Id-ext\,O2O$	**50.04**	26.31	**47.34**	**24.61**	**20.14**
$Id-ext\,M2O_{n=2}$	47.00	23.37	44.23	22.22	17.99

Two-to-one extractor-paraphraser model without reinforcement [$\mathbf{M2O_{n=2}\,(w/o\,RL)}$]: A specific case of the proposed many-to-one paraphrasing where $n = 2$. This specific model comprises of only the extractor and paraphraser modules.

Two-to-one extractor-paraphraser model [$\mathbf{M2O_{n=2}+\,RL_X}$]: A specific case of the proposed many-to-one paraphrasing where $n = 2$; including X as the reward function for reinforced extractor (X is either ROUGE-L or the reward function defined in Eq. 5) and beam search.

Ideal extractor-abstractor model [$\mathbf{Id-ext\ EXT-ABS}$]: To determine the performance of the abstractor component, the $EXT-ABS$ baseline is evaluated with the assumption that the extractor is ideal, subsequently feeding the exemplary-extracted sentences generated for the test data using ROUGE-L score to the paraphraser directly.

Ideal extractor-paraphraser model [$\mathbf{Id-ext\ O2O}$]: Similar to the $Id-ext\,EXT-ABS$ setup, the 'n-to-1' paraphraser is evaluated with $n = 1$, given that the extractor performs ideally (generating exemplary-extracted sentences as proposed in Eq. 4).

Ideal extractor-paraphraser model [$\mathbf{Id-ext\ M2O_{n=2}}$]: The 'n-to-1' paraphraser with $n = 2$ and the exemplary-extracted sentences assumed as the extractor (generating exemplary-extracted sentences as proposed in Eq. 4).

5 Results

Results of different baselines and the proposed approach are discussed in this section. Table 1 illustrates that our proposed techniques perform better than the rest of the systems. We have used ROUGE-1, ROUGE-2, ROUGE-L (Lin, 2004), METEOR (Banerjee and Lavie, 2005) and word mover similarity (WMS) (Clark et al., 2019) as evaluation metrics. We believe that ROUGE as an evaluation metric is incapable of judging the quality of an abstractive summary due to its emphasis on syntactic overlap over semantic overlap (Liu et al., 2016; Clark et al., 2019; Novikova et al., 2017). To overcome this, we have also used WMS as an evaluation metric.

5.1 Semantic information overlap

It is noticed that the proposed paraphraser in model $O2O\ (w/o\,RL)$ outperforms the abstractor from $EXT-ABS\ (w/o\,RL)$ in terms of ROUGE scores, while scoring marginally less in terms of METEOR. The extractor counterparts $EXT-ABS\,(Ext\,only)$ and $O2O\,(Ext\,only)$

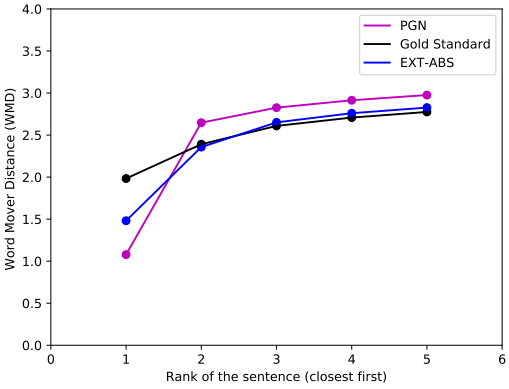


Figure 2: Average sentence distance (word mover distance) scores for most similar sentences.

also portray a similar tendency, with a wider gap in METEOR scores. We also observe that in the reinforced extractor models, the $O2O + RL_X$ setting surpasses the $EXT - ABS + RL_X$ setting in almost all the metrics[8]. A similar trend is also observed in the ideal extractor experiments, where the $Id - ext\, O2O$ model beats the $Id - ext\, EXT - ABS$ model in WMS while keeping other evaluation scores comparable. The above mentioned observations illustrate the true capabilities of using semantic overlap based exemplary-extracted sentences.

Keeping the main model same as the $EXT - ABS$ framework, and changing the reward function from ROUGE-L to WMS (Eq. 5), it is observed that the latter ($EXT - ABS\, RL_{WMS}$) attains significantly better METEOR and WMS scores, while maintaining comparable ROUGE scores. However, the true capabilities of the WMS reward function (Eq. 5) come into play when we attach it with our extractor-paraphraser framework; $O2O\, RL_{WMS}$ bests every other models in terms of WMS, while keeping ROUGE and METEOR comparable with the best attained scores.

One critical observation is that the reward function introduces a bias in the evaluation process. It can be clearly observed from the fact that the model $EXT - ABS + RL_{ROUGE}$ obtains better ROUGE scores while it pales in comparison to $EXT - ABS + RL_{WMS}$ in terms of WMS. Since we stress that semantic information overlap is more significant than the syntactic overlap, we believe that WMS is better suited for the evaluation task as well as a better choice for the reward function.

[8]Here $X \in \{ROUGE, WMS\}$, which remains same when comparing the two models

It is established by the fact that the models using WMS as the reward function attain comparable ROUGE scores as well (while the reverse is not true), indicating that incorporating semantic information can assist in capturing syntactic information as well. Examples of generated summaries for the $EXT - ABS\, RL_{ROUGE}$ and $O2O\, RL_{WMS}$ models are illustrated in Fig. 3. An important observation in the generated summaries is that the former model produces the name "shao li" which is not present in the input document or the gold standard summary, whereas this mistake is avoided by the $O2O\, RL_{WMS}$ model.

5.2 Summarization vs paraphrasing

Theoretically the $M2O_{n=2}$ setting should surpass the $O2O$ setting, since the former has extra input information at the paraphraser stage that the latter lacks. However, it is observed that this does not happen; in actuality, the $M2O_{n=2}$ model is outperformed by the $O2O$ model in all aspects (Table 1). After manual scrutiny of *exemplary-extracted sentence* pairs fed to the paraphraser and the generated sentences, it is observed that the expected accumulation of information does not take place. To quantify this observation, WMD based overlap of information is computed between the $exemplary - extracted\, sentence$ and the generated sentence. It is discovered that on average, the more similar sentence has a WMD of **1.775** while the other one obtains a value of **2.894**, illustrating the inability of the paraphraser to combine information across the two sentences into one.

Hence, we hypothesize that the PGN model intrinsically paraphrases one input sentence to generate the corresponding sentence in the generated summary, innately mimicking the extractor paraphraser behaviour. An experiment is formulated to evince the truth of this proposed hypothesis. For this experiment, three different collections of documents are used: 1) ground truth summary, 2) summaries generated by PGN (See et al., 2017), and 3) summaries generated by the $EXT - ABS$ model (Chen and Bansal, 2018), all corresponding to the data in the test set. For every summary sentence, its semantic overlap (using WMD) is computed with every document sentence and distances of the closest α document sentences are reported in an increasing order. The results of this experiment can be seen in Fig. 2, and it is noticed that the average gap between similarities with respect to first

and second most similar sentences is the highest in the case of the PGN model (approx. 1.57), while the value is around 0.41 for the gold standard summaries. The curve for gold standard has a steady slope, where as the slope for PGN rises quickly from the first point to the second point, and then follows a constant growth afterwards, illustrating a very high similarity with the most similar sentence, thus endorsing the hypothesis that the PGN intrinsically paraphrases one input sentence to generate one sentence in final summary.

Gold Standard:
tong shao, 20, was an international student from china attending iowa state university. her body was found in the trunk of her car in iowa city on september 26. police believe it had been for three weeks. she died of blunt force trauma and asphyxiation. her boyfriend, xiangnan li, 23, was the last to see her, but flew to china on september 8, before shao was officially missing. according to tong 's father, an arrest warrant has now been issued. however li has disappeared.
EXT-ABS + RL_{ROUGE}:
police are issuing a warrant for the girl's boyfriend. police found her body stuffed in the trunk of her toyota camry. shao li, 23, was listed as a person of interest in the case. tong shao was a chemical engineering student at iowa state university. xiangnan li, 23, found murdered in september after going missing. the girl 's body was discovered in iowa last year. tong shao, 20, was found dead in her car for three weeks.
O2O + RL_{WMS}:
tong shao went missing in september 2014. police found her body stuffed in the trunk of her toyota camry. shao's boyfriend, xiangnan li, 23, was listed as a person of interest. police are issuing a warrant for the girl's boyfriend. a 20-year-old daughter was found murdered in iowa last year.

Figure 3: Example of generated summaries for $EXT-ABS+RL_{ROUGE}$ model, $O2O+RL_{WMS}$. The text in red denotes the novel information that is not present in the gold summary, the text in green denotes the information overlap that is present exclusively in the generated summary and the text in blue denotes the information covered in all three scenarios.

5.3 Error Analysis

As the results illustrate in Table 1, overall the one-to-one setting of the proposed n-to-one paraphraser performs better than the two-to-one setting, revealing a major blockade of the existing sequential frameworks - the incapability of combining multiple sentences into one, deviating from the ideal notion of 'abstraction'. We also observe that the extractors outperform their corresponding $(w/o\,RL)$ counterparts, and are competing with the complete models as well to some extent, portraying the difficulty of abstractive summarization over sentence extraction, along with the inability of current evaluation metrics like ROUGE and METEOR to focus on semantic information over syntactic subtleties. As elucidated by the ideal extractor experiments, the paraphraser has great potential for even achieving state-of-the-art results in the summarization task, provided the extractor works ideally. Our future work would focus on mollifying the loss introduced at the extraction step, and obtaining a better harmony in the extractor-paraphraser network as a whole.

6 Conclusion

In this paper, we propose the extractor-paraphraser model, that comprises of the n-to-one paraphraser and a novel word mover similarity based reward function. We show that the semantic overlap based techniques surpass the strong baselines that rely on syntactic information overlap. We also develop experiments to unveil that the state-of-the-art pointer-generator network (PGN) indeed paraphrases input sentences intrinsically, and is unable to merge two sentences into one agglomerate sentence when explicitly conditioned to do so, changing our perception of sequence-to-sequence abstractive summarization models. We show that the existing works are still nowhere close to mimicking a true human summary, portraying the difficulty of true abstraction over its simplified formulation of extracting and then paraphrasing.

Acknowledgement: Dr. Sriparna Saha would like to acknowledge the support of Early Career Research Award of Science and Engineering Research Board (SERB) of Department of Science and Technology, India to carry out this research.

References

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Siddhartha Banerjee, Prasenjit Mitra, and Kazunari Sugiyama. 2015. Multi-document abstractive summarization using ilp based multi-sentence compression. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. *arXiv preprint arXiv:1805.11080*.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98.

Elizabeth Clark, Asli Celikyilmaz, and Noah A Smith. 2019. Sentence mover's similarity: Automatic evaluation for multi-sentence texts. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2748–2760.

Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. 2018. Banditsum: Extractive summarization as a contextual bandit. *arXiv preprint arXiv:1809.09672*.

Yijun Duan and Adam Jatowt. 2019. Across-time comparative summarization of news articles. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 735–743. ACM.

Dimitrios Galanis, Gerasimos Lampouras, and Ion Androutsopoulos. 2012. Extractive multi-document summarization with integer linear programming and support vector regression. In *Proceedings of COLING 2012*, pages 911–926.

Mahak Gambhir and Vishal Gupta. 2017. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, 47(1):1–66.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: A graph based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 340–348, Beijing, China. Coling 2010 Organizing Committee.

Yang Gao, Christian M Meyer, Mohsen Mesgar, and Iryna Gurevych. 2019. Reward learning for efficient reinforcement learning in extractive document summarisation. *arXiv preprint arXiv:1907.12894*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Anubhav Jangra, Adam Jatowt, Mohammad Hasanuzzaman, and Sriparna Saha. 2020a. Text-image-video summary generation using joint integer linear programming. In *European Conference on Information Retrieval*, pages 190–198. Springer.

Anubhav Jangra, Sriparna Saha, Adam Jatowt, and Mohammad Hasanuzzaman. 2020b. Multi-modal summary generation using multi-objective optimization. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 1745–1748.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 68–73. ACM.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

Linqing Liu, Yao Lu, Min Yang, Qiang Qu, Jia Zhu, and Hongyan Li. 2018. Generative adversarial network for abstractive text summarization. In *Thirty-second AAAI conference on artificial intelligence*.

Rada Mihalcea. 2004. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 170–173.

Rada Mihalcea and Hakan Ceylan. 2007. Explorations in automatic book summarization. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 380–389.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for nlg. *arXiv preprint arXiv:1707.06875*.

Chris D Paice. 1990. Constructing literature abstracts by computer: techniques and prospects. *Information Processing & Management*, 26(1):171–186.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Samuel Pecar. 2018. Towards opinion summarization of customer reviews. In *Proceedings of ACL 2018, Student Research Workshop*, pages 1–8.

Naveen Saini, Sriparna Saha, Pushpak Bhattacharyya, and Himanshu Tuteja. 2020. Textual entailment–based figure summarization for biomedical articles. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 16(1s):1–24.

Naveen Saini, Sriparna Saha, Dhiraj Chakraborty, and Pushpak Bhattacharyya. 2019a. Extractive single document summarization using binary differential evolution: Optimization of different sentence quality measures. *PloS one*, 14(11):e0223477.

Naveen Saini, Sriparna Saha, Anubhav Jangra, and Pushpak Bhattacharyya. 2019b. Extractive single document summarization using multi-objective optimization: Exploring self-organized differential evolution, grey wolf optimizer and water cycle algorithm. *Knowledge-Based Systems*, 164:45–67.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. *CoRR*, abs/1704.04368.

Liqiang Xiao, Lu Wang, Hao He, and Yaohui Jin. 2020. Copy or rewrite: Hybrid summarization with hierarchical reinforcement learning. In *AAAI*, pages 9306–9313.

Jin-ge Yao, Xiaojun Wan, and Jianguo Xiao. 2017. Recent advances in document summarization. *Knowledge and Information Systems*, 53(2):297–336.

Yong Zhang, Meng Joo Er, Rui Zhao, and Mahardhika Pratama. 2016. Multiview convolutional neural networks for multidocument extractive summarization. *IEEE transactions on cybernetics*, 47(10):3230–3242.

Yuhao Zhang, Daisy Yi Ding, Tianpei Qian, Christopher D Manning, and Curtis P Langlotz. 2018. Learning to summarize radiology findings. *arXiv preprint arXiv:1809.04698*.

Junnan Zhu, Yu Zhou, Jiajun Zhang, Haoran Li, Chengqing Zong, and Changliang Li. 2020. Multimodal summarization with guidance of multimodal reference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9749–9756.

Li Zhuang, Feng Jing, and Xiao-Yan Zhu. 2006. Movie review mining and summarization. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 43–50.

*Thamizhi*UDp: A Dependency Parser for Tamil

Kengatharaiyer Sarveswaran
University of Moratuwa / Sri Lanka
sarvesk@uom.lk

Gihan Dias
University of Moratuwa / Sri Lanka
gihan@uom.lk

Abstract

This paper describes how we developed a neural-based dependency parser, namely *Thamizhi*UDp, which provides a complete pipeline for the dependency parsing of the Tamil language text using Universal Dependency formalism. We have considered the phases of the dependency parsing pipeline and identified tools and resources in each of these phases to improve the accuracy and to tackle data scarcity. *Thamizhi*UDp uses Stanza for tokenisation and lemmatisation, *Thamizhi*POSt and *Thamizhi*Morph for generating Part of Speech (POS) and Morphological annotations, and uuparser with multilingual training for dependency parsing. *Thamizhi*POSt is our POS tagger, which is based on the Stanza, trained with Amrita POS-tagged corpus. It is the current state-of-the-art in Tamil POS tagging with an F1 score of 93.27. Our morphological analyzer, *Thamizhi*Morph is a rule-based system with a very good coverage of Tamil. Our dependency parser *Thamizhi*UDp was trained using multilingual data. It shows a Labelled Assigned Score (LAS) of 62.39, 4 points higher than the current best achieved for Tamil dependency parsing. Therefore, we show that breaking up the dependency parsing pipeline to accommodate existing tools and resources is a viable approach for low-resource languages.

1 Introduction

Applying neural-based approaches to Tamil, like other Indic languages, is challenging due to a lack of quality data (Bhattacharyya et al., 2019), and the language's structure (Sarveswaran and Butt, 2019; Butt, Miriam, Rajamathangi, S. and Sarveswaran, K., 2020). Although there is a large volume of electronic unstructured/partially-structured text available on the Internet, not many language processing tools are publicly available for even fundamental tasks like part of speech (POS) tagging or parsing. Nowadays, neural-based approaches are the state of the art for most natural language processing tasks. These approaches require a significant amount of quality data for training and evaluation. On the other hand techniques like transfer learning, and multilingual learning may be used to overcome data scarcity. This paper discusses how we developed a neural-based dependency parser for Tamil with the aid of data orchestration and multilingual training.

2 Background and Motivation

Tamil is a Southern Dravidian language spoken by more than 80 million people around the world. However, it still lacks enough tools and quality annotated data to build good Natural Language Processing (NLP) applications.

2.1 Universal Dependency Treebank

Treebanks are a collection of texts with various levels of annotations, including Part of Speech (POS) and morpho-syntactic annotations. There are different formalisms used to mark syntactic annotations (Marcus et al., 1993; Böhmová et al., 2003; Nivre et al., 2016; Kaplan and Bresnan, 1982). Among the available formalisms, the dependency grammar formalism is useful for languages like Tamil which are morphologically rich, and whose word order is relatively variable and less bound (Bharati et al., 2009).

The Universal Dependency formalism (Nivre et al., 2016) is nowadays used widely to create Universal Dependency Treebanks (UD) with annotations. The current release of UDv2.7 has 183 annotated treebanks of various sizes from 104 languages (Zeman et al., 2020). UD captures information such as Parts of Speech (POS), morphological features, and syntactic relations in the form of dependencies. All these annotations are defined with multilingual language processing in mind, and the

Proceedings of the 17th International Conference on Natural Language Processing, pages 200–207
Patna, India, December 18 - 21, 2020.

present format used to specify the annotation is called CoNLL-U format.[1]

There are only three Indic languages, namely, Hindi, Urdu, and Sanskrit that have relatively large datasets 375K, 138K, and 28K tokens, respectively in UDv2.7. All the other six Indic languages, including Tamil and Telugu, have less than 12K tokens in UDv2.7.

2.2 Tamil Universal Dependency Treebanks

Tamil has been included in UD treebank releases since 2015. Initially it was populated from the Prague Style Tamil treebank by (Ramasamy and Žabokrtský, 2012), and since then the dataset has been part of the UD without much alterations or corrections. Tamil TTB in UDv2.6 has some inaccuracies, and inconsistencies. For instance, numbers are marked as NUM and ADJ, while only the former tag is correct. The first author of this paper has corrected some of these issues and made it available in UDv2.7. However, there are still more issues that need to be solved. Tamil TTB in UDv2.7 has altogether 600 sentences for training, development and testing. In (Zeman et al., 2020), there is another Tamil treebank with 536 sentences, namely MWTT, which has been newly added. MWTT is based on the Enhanced Universal Dependency[2] annotation, where complex concepts like elision, relative clauses, propagation of conjuncts, raising and control constructions, and extended case marking are captured. Therefore, there are slight variations in TTB and MWTT. Further, MWTT has very short sentences, while TTB has relatively very longer ones. In this paper, we have mainly used and discussed Tamil TTB.

2.3 Dependency parsers

A Dependency parser is a type of syntactic parser which is useful to elicit lexical, morphological, and syntactic features, and the inter-connections of tokens in a given sentence. Linguistically, this would be useful for syntactic analyses, and comparative studies. Computationally, this is a key resource for natural language understanding (Dozat and Manning, 2018). Different approaches are employed when developing dependency parsers. However, neural-based parsers are the latest state of the art.

There are several off-the-shelf neural-based parsers available that are built around Universal Dependency Treebanks (UD), including Stanza (Qi et al., 2020), and uuparser (de Lhoneux et al., 2017) and its derivatives. Both of these are open source tools. Stanza is a Python NLP library which includes a multilingual neural NLP pipeline for dependency parsing using Universal Dependency formalism. uuparser is a tool developed specifically for UD parsing. These neural-based tools need large amount of quality data on which to be trained.

On the other hand, several approaches are being used to overcome the issue with data scarcity, including multilingual training. There is an attempt to create a multilingual parsing for several low-resource languages, and it is reported that multilingual training significantly improves the parsing accuracy of low-resource languages (Smith et al., 2018).

Figure 1: Phases of the Parsing Pipeline
Image source: https://stanfordnlp.github.io/stanza

3 *Thamizhi*UDp

By considering all available resources and approaches as outlined in section 2, we decided to develop a Universal Dependency parser (UDp) for Tamil called *Thamizhi*UDp using existing open source tools, namely Stanza, *Thamizhi*Morph, and uuparser. However, since we do not have enough data to train a neural-based parser end-to-end, we have broken up the pipeline to different phases. We have then orchestrated data from different sources for each of these phases, and used different tools in different phases, as shown in Table 1. The following sub-sections discuss each of the stages of the pipeline, and how we went about developing them.

Our dependency parsing pipeline has several stages as shown in Figure 1. As mentioned, we used different datasets and tools, as shown in Table 1, in the different stages of the pipeline. Currently,

[1] https://universaldependencies.org/format.html
[2] https://universaldependencies.org/u/overview/enhanced-syntax.html

Step	Tool	Dataset
Tokenisation	Stanza	Tamil UDT
Multi-word tokeniser	Stanza	Tamil UDT
Lemmatisation	Stanza	Tamil UDT
POS tagging	*Thamizhi*POSt	Amrita Data
Morphological tagging	*Thamizhi*Morph	Rule-based
Dependency parsing	uuparser	UDT of various languages

Table 1: *Thamizhi*UDp process pipeline

Stanza does not have support for multilingual training. Therefore, for dependency parsing, we used uuparser with the multilingual training. Each of the phases within the pipeline is explained in the following respective sub-sections.

3.1 Tokenisation

First, the given texts have been Unicode normalised, and then tokenised, and broken up in to sentences. We developed a script[3] to do Unicode normalisation. Because of different input methods or other reasons, at times the same surface form of a character has been stored using different Unicode sequences. Therefore, this needed to be normalised, otherwise, a computer would consider them as different characters. Once this normalisation was done, we moved on to tokenisation. To do this, we trained Stanza with the texts available in TTB. During this phase, punctuations were separated from words, and the given texts were broken in to sentences.

3.2 Multi-word tokenisation using Stanza

After the initial tokenisation, syntactically compound words or multi-word tokens were broken into syntactic units as proposed by the UD guidelines,[4] so that syntactic dependencies can be marked precisely. Syntactically compound constructions are common in Tamil. For instance, words with *-um* clitic will be tokenised, like *naanum* =⟩ *naan+um* 'I+and', so that coordinating conjunctive dependency can be shown easily. In the current TTB UDv2.7, there are 520 instances of multi-words found among 400 sentences in the training set. We used this TTB training set to train our multi-word tokeniser using Stanza. However, multi-word tokenisations are not properly divided in TTB. We are in the process of improving this multi-word tokeniser with the use of more data.

3.3 Lemmatisation using Stanza

UD Treebanks also have lemmas marked in their CoNLL-U format annotation This is useful for language processing applications, such as a Machine Translator. We trained Stanza using the TTB UDv2.6 to do lemmatisation. However, the current TTB has several inaccuracies in identifying lemmas, specifically due to improper multi-word tokenisation. Since a lemma is identified for multi-word tokenised words, multi-word tokenisation has an effect on lemmatisation. Since we are still in the process of improving our multi-word tokeniser, lemmatisation will also be improved in the future.

3.4 POS tagging using *Thamizhi*POSt

Part of Speech (POS) tagging is an important phase in the parsing process where each word in a sentence is assigned with its POS tag (or lexical category) information. Several attempts have been made to define POS tagsets for Tamil, based on different theories, and level of granularity; (Sarveswaran and Mahesan, 2014) gives an account of different tagsets. Among these, Amrita (Anand Kumar et al., 2010) and BIS[5] are two popular tagsets. In addition to tagsets, Amrita and BIS POS tagged data are also available. The corpus[6] which is tagged using BIS tagset is taken from a historical novel, while the corpus tagged using Amrita is taken from news websites. Further more we found that Amrita's data is cleaner and there is more consistency when it comes to POS tagging. We also harmonised the tags found in the BIS, Amrita, and UPOS[7] tagsets.

Though there have been several attempts to develop a POS tagger for Tamil, there are not available, or have not given convincing results. Moreover, only a few neural-based approaches for Tamil POS tagging have been developed. Therefore, we decided to develop a POS tagger, namely *Thamizhi*POSt, using Stanza, and to publish it as an open source tool. We used the corpus tagged using Amrita's POS tagged corpus to train this tagger. The development process is outlined briefly below.

[3] https://github.com/sarves/thamizhi-validator
[4] https://universaldependencies.org/u/overview/tokenization.html
[5] http://www.tdil-dc.in/tdildcMain/articles/134692DraftPOSTagstandard.pdf
[6] http://www.au-kbc.org/nlp/corpusrelease.html
[7] https://universaldependencies.org/u/pos/all.html

Neural-based POS taggers	F1 Score
PVS and Karthik (2007)	87.0*
Mokanarangan et al. (2016)	87.4
Qi et al. (2020)	82.6**
***Thamizhi*POSt**	**93.27**

Table 2: Scores of neural-based POS taggers for Tamil POS tagging
*github.com/avineshpvs/indic_tagger
**stanfordnlp.github.io/stanza/performance.html

First we mapped Amrita's 32 POS tags to Universal POS (UPOS); see Table 4 in Appendix A for the mapping of Amrita-UPOS mapping. In doing so, we converted the annotations from Amrita POS tags to UPOS tags. We then divided Amrita POS tagged corpus of 17K sentences in to 11K, 5K and 1K sentences for training, development, and testing, respectively. Thereafter, we converted these datasets in CoNLL-U format so that it could then be fed to Stanza. Following that we trained and evaluated *Thamizhi*POSt, which is a Stanza instance that has been trained on Amrita's data. During the training, we also used fastText model (Bojanowski et al., 2016) to capture the context in POS tagging, as specified in Stanza.

We also evaluated *Thamizhi*POSt using Tamil UDv2.6 test data. The F1 score of the evaluation was 93.27, which is higher than the results reported for existing neural-based POS taggers, as shown Table 2.

3.5 Morphological tagging

We used an open source morphological analyser called (Sarveswaran et al., 2019, 2018),[8] which we developed as part of our project on computational grammar for Tamil, to generate morphological features according to the UD specification.[9] Since we have developed this rule-based analyser for grammar development purposes, this gives us a very detailed analysis for each given word. We used *Thamizhi*Morph in the process. At this stage, we fed the tokenised, lemmatised and POS tagged data in the CoNLL-U format to *Thamizhi*Morph to do the morphological analyses.

As a morphological analyser, *Thamizhi*Morph gives us all the possible morphological analyses for a given word. In addition to the morphological analysis, it also gives us the POS tag information, and lemma information. When the lemma of a given surface form in not found in the *Thamizhi*Morph lexicon, it uses a rule-based guesser to predict the lemma; sometimes this fails too, especially when there is a foreign word.

However, for our parsing purpose, we wanted to get the single correct morphological analysis based on the context. This was challenging. To tackle this challenge, we used a disambiguation process to generate a single analysis. However, we still failed at times, since we especially get multiple analyses because of the way some people write. When this was the case, we manually picked the correct analysis, even after our disambiguation process. We are now in the process of training a Stanza based morphological analyser using the data generated by *Thamizhi*Morph. We hope this will improve the robustness, especially when there are out of vocabulary tokens.

3.6 Dependency parsing

When we looked for a Dependency parser, we found that none existed that were specifically trained for Tamil. For TTB test data, in their default configurations the off-the-shelf Stanza and uuparser give the Labelled Assigned Score (LAS) of 57.64 and 55.76, respectively.

We wanted to improve the accuracy, however, we could not find any datasets with dependency annotations, other than TTB UDv2.6 at the time of development. To overcome this data scarcity, we tried multilingual training for Tamil along with Hindi HDTB,[10] Turkish,[11] Arabic,[12] and Telugu,[13] which we found would be relevant, available in UDv2.6. We did this multilingual training using uuparser. The experiment gave us some good results, when we compared this with what was reported by Stanza or uuparser as shown in Table 3.

As in Table 3, we got a LAS of 62.39 when training with Hindi HDTB UDv2.6, but, surprisingly, not when training with Telugu, which is also a Dravidian language like Tamil. We trained the tagger with the whole Telugu, and Hindi treebanks along with Tamil. However, the score was lesser than what we got when we trained it with Hindi

[8] https://github.com/sarves/thamizhi-morph
[9] https://universaldependencies.org/u/feat/all.html
[10] https://github.com/UniversalDependencies/UD_Hindi-HDTB/tree/master
[11] https://github.com/UniversalDependencies/UD_Turkish-IMST/tree/master
[12] https://github.com/UniversalDependencies/UD_Arabic-PADT/tree/master
[13] https://github.com/UniversalDependencies/UD_Telugu-MTG/tree/master

Languages (# of sent.)	Accuracy(LAS)
with Telugu (100)	58.91
with Telugu (1050)	59.22
with Hindi (1600)	**62.39**
with Telugu (100) and Arabic (100)	58.04
with Telugu (100) and Turkish (100)	58.43
with Telugu (100) and Hindi (100)	59.07

Table 3: LAS of Multilingual parsing

data. For all these experiments, we used the Tamil testing set available in TTB UDv2.6.

4 Discussion

Tamil TTB has not undergone any major revisions or corrections since its initial release. It has several issues, in POS tagging, multi-word tokenisation, and dependency tagging. Altogether we only have 600 sentences for training, development, and testing; some of these sentences are very long. All these made the training of a UD parser a difficult task. We tried to overcome some of these issues using other data, and tools available online. However, we still depend on this dataset for some part of the training, such as for dependency parsing.

Only one treebank, Telugu MTG UDv2.6, which is the closest to Tamil in terms of linguistic structures, is available as of today. We observed that Telugu UDv2.6 is small in size. That only has around 1050 sentences compared to Hindi HDTB UDv2.6. Moreover, Telugu has very short sentences without any morphological feature information. On the other hand, some sentences in TTB in UDv2.6 has up to 40 tokens. Because of all these varied factors we could not achieve much improvement when use Telugu MTG UDv2.6 in multilingual training. However, Hindi, which belongs to a different language family, showed better performance when used for Multilingual training. We have additionally noticed that the accuracy of the dependency parsing also improved when we increased the Hindi data size during the training.

Another challenge we have faced was finding quality test data or benchmark datasets for evaluation. In the current practice, everyone tests their tools using their own dataset to evaluate. Therefore, it is always a challenging task to reproduce or compare results. In our case, for dependency parsing, we used the UD test data. However, it is not a clean and error free dataset for evaluation. For this reason, we have now started working on a Tamil dependency treebank which can soon be used as an evaluation dataset.

We used our personal computers without any Graphical Processing Units (GPU) to carry out all these experiments. However, high performance computing resources will save time, and we might need to go for such resources when we increase the size of datasets.

5 Conclusion

e have implemented a Universal Dependency parser for Tamil, *Thamizhi*UDp, which annotates a Tamil sentence with POS, Lemma, Morphology, and Dependency information in CoNLL-U format. We have developed a parsing pipeline using several open source tools and datasets to overcome data scarcity. We have also used multilingual training to overcome the scarecity of dependency annotated data. *Thamizhi*POSt, a POS tagger for Tamil, has been implemented using Stanza and the Amrita POS tagged dataset. *Thamizhi*POSt outperforms existing neural-based POS taggers, and gives an F1 score of 93.27. Further, we obtained the best accuracy of LAS 62.39 for dependency parsing in a multilingual training setting with Hindi HDTB, using uuparser. More importantly, we have made our tools *Thamizhi*POSt,[14,15] *Thamizhi*Morph,[16,17] and *Thamizhi*UDp[18,19] along with relevant models, datasets, and scripts available open source for others to use and extend upon.

Acknowledgements

We would like express our appreciation to Maris Camilleri from the University of Essex for her support in language editing, and three anonymous reviewers for their valuable comments and inputs to improve this menu script.

This research was supported by the Accelerating Higher Education Expansion and Development (AHEAD) Operation of the Ministry of Higher Education, Sri Lanka funded by the World Bank.

[14] http://nlp-tools.uom.lk/thamizhi-pos
[15] https://github.com/sarves/thamizhi-pos/
[16] http://nlp-tools.uom.lk/thamizhi-morph
[17] https://github.com/sarves/thamizhi-morph/
[18] http://nlp-tools.uom.lk/thamizhi-udp
[19] https://github.com/sarves/thamizhi-udp/

References

M Anand Kumar, V Dhanalakshmi, KP Soman, and S Rajendran. 2010. A sequence labeling approach to morphological analyzer for Tamil language. *International Journal on Computer Science and Engineering (IJCSE)*, 2(06):1944–195.

Akshar Bharati, Mridul Gupta, Vineet Yadav, Karthik Gali, and Dipti Misra Sharma. 2009. Simple parser for Indian languages in a dependency framework. In *Proceedings of the Third Linguistic Annotation Workshop (LAW III)*, pages 162–165.

Pushpak Bhattacharyya, Hema Murthy, Surangika Ranathunga, and Ranjiva Munasinghe. 2019. Indic language computing. *Communications of the ACM*, 62(11):70–75.

Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2003. The Prague Dependency Treebank. In *Treebanks*, pages 103–127. Springer.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Butt, Miriam, Rajamathangi, S. and Sarveswaran, K. 2020. Mixed Categories in Tamil via Complex Categories. In *Proceedings of the LFG20 Conference*, Stanford. CSLI Publications.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490, Melbourne, Australia. Association for Computational Linguistics.

Ron Kaplan and Joan Bresnan. 1982. Lexical functional grammar: a formal system for grammatical representation. The Mental Representation of Grammatical Relations. *J. Bresnan. Cambridge, MA: MIT Press*, pages 173–281.

Miryam de Lhoneux, Yan Shao, Ali Basirat, Eliyahu Kiperwasser, Sara Stymne, Yoav Goldberg, and Joakim Nivre. 2017. From raw text to universal dependencies-look, no tags! In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 207–217.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

T Mokanarangan, T Pranavan, U Megala, N Nilusija, Gihan Dias, Sanath Jayasena, and Surangika Ranathunga. 2016. Tamil morphological analyzer using support vector machines. In *International Conference on Applications of Natural Language to Information Systems*, pages 15–23. Springer.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666.

Avinesh PVS and G Karthik. 2007. Part-of-speech tagging and chunking using conditional random fields and transformation based learning. *Shallow Parsing for South Asian Languages*, 21:21–24.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. *arXiv preprint arXiv:2003.07082*.

Loganathan Ramasamy and Zdeněk Žabokrtský. 2012. Prague dependency style treebank for Tamil. In *Proceedings of Eighth International Conference on Language Resources and Evaluation (LREC 2012)*, pages 1888–1894, İstanbul, Turkey.

K Sarveswaran, Gihan Dias, and Miriam Butt. 2018. ThamizhiFST: A Morphological Analyser and Generator for Tamil Verbs. In *2018 3rd International Conference on Information Technology Research (ICITR)*, pages 1–6. IEEE.

K Sarveswaran, Gihan Dias, and Miriam Butt. 2019. Using meta-morph rules to develop morphological analysers: A case study concerning Tamil. In *Proceedings of the 14th International Conference on Finite-State Methods and Natural Language Processing*, pages 76–86, Dresden, Germany. Association for Computational Linguistics.

K Sarveswaran and S Mahesan. 2014. Hierarchical Tag-set for Rule-based Processing of Tamil Language. *International Journal of Multidisciplinary Studies (IJMS)*, 1(2):67–74.

Kengatharaiyer Sarveswaran and Miriam Butt. 2019. Computational Challenges with Tamil Complex Predicates. In *Proceedings of the LFG19 Conference, Australian National University*, pages 272–292, Stanford. CSLI Publications.

Aaron Smith, Bernd Bohnet, Miryam de Lhoneux, Joakim Nivre, Yan Shao, and Sara Stymne. 2018. 82 treebanks, 34 models: Universal Dependency parsing with multi-treebank models. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 113–123, Brussels, Belgium. Association for Computational Linguistics.

Daniel Zeman, Joakim Nivre, Mitchell Abrams, Elia Ackermann, Noëmi Aepli, Hamid Aghaei, Željko Agić, Amir Ahmadi, Lars Ahrenberg, Chika Kennedy Ajede, Gabrielė Aleksandravičiūtė,

Ika Alfina, Lene Antonsen, Katya Aplonova, Angelina Aquino, Carolina Aragon, Maria Jesus Aranzabe, Hórunn Arnardóttir, Gashaw Arutie, Jessica Naraiswari Arwidarasti, Masayuki Asahara, Luma Ateyah, Furkan Atmaca, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Keerthana Balasubramani, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, Victoria Basmov, Colin Batchelor, John Bauer, Seyyit Talha Bedir, Kepa Bengoetxea, Gözde Berk, Yevgeni Berzak, Irshad Ahmad Bhat, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Agnė Bielinskienė, Kristín Bjarnadóttir, Rogier Blokland, Victoria Bobicev, Loïc Boizou, Emanuel Borges Völker, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Kristina Brokaitė, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Tatiana Cavalcanti, Gülşen Cebiroğlu Eryiğit, Flavio Massimiliano Cecchini, Giuseppe G. A. Celano, Slavomír Čéplö, Savas Cetin, Özlem Çetinoğlu, Fabricio Chalub, Ethan Chi, Yongseok Cho, Jinho Choi, Jayeol Chun, Alessandra T. Cignarella, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Mehmet Oguz Derin, Elvis de Souza, Arantza Diaz de Ilarraza, Carly Dickerson, Arawinda Dinakaramani, Bamba Dione, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Hanne Eckhoff, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Olga Erina, Tomaž Erjavec, Aline Etienne, Wograine Evelyn, Sidney Facundes, Richárd Farkas, Marília Fernanda, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Kazunori Fujita, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Sebastian Garza, Fabrício Ferraz Gerardi, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta González Saavedra, Bernadeta Griciūtė, Matias Grioni, Loïc Grobol, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Tunga Güngör, Nizar Habash, Hinrik Hafsteinsson, Jan Hajič, Jan Hajič jr., Mika Hämäläinen, Linh Hà Mỹ, Na-Rae Han, Muhammad Yudistira Hanifmuti, Sam Hardwick, Kim Harris, Dag Haug, Johannes Heinecke, Oliver Hellwig, Felix Hennig, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Eva Huber, Jena Hwang, Takumi Ikeda, Anton Karl Ingason, Radu Ion, Elena Irimia, Ọlájídé Ishola, Tomáš Jelínek, Anders Johannsen, Hildur Jónsdóttir, Fredrik Jørgensen, Markus Juutinen, Sarveswaran K, Hüner Kaşıkara, Andre Kaasen, Nadezhda Kabaeva, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Tolga Kayadelen, Jessica Kenney, Václava Kettnerová, Jesse Kirchner, Elena Klementieva, Arne Köhn, Abdullatif Köksal, Kamil Kopacewicz, Timo Korkiakangas, Natalia Kotsyba, Jolanta Kovalevskaitė, Simon Krek, Parameswari Krishnamurthy, Sookyoung Kwak, Veronika Laippala, Lucia Lam, Lorenzo Lambertino, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phng Lê H`ông, Alessandro Lenci, Saran Lertpradit, Herman Leung, Maria Levina, Cheuk Ying Li, Josie Li, Keying Li, Yuan Li, KyungTae Lim, Krister Lindén, Nikola Ljubešić, Olga Loginova, Andry Luthfi, Mikko Luukko, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Hiroshi Matsuda, Yuji Matsumoto, Ryan McDonald, Sarah McGuinness, Gustavo Mendonça, Niko Miekka, Karina Mischenkova, Margarita Misirpashayeva, Anna Missilä, Cătălin Mititelu, Maria Mitrofan, Yusuke Miyao, AmirHossein Mojiri Foroushani, Amirsaeid Moloodi, Simonetta Montemagni, Amir More, Laura Moreno Romero, Keiko Sophie Mori, Shinsuke Mori, Tomohiko Morioka, Shigeki Moro, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Robert Munro, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Mariam Nakhlé, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lng Nguy˜ên Thị, Huy`ên Nguy˜ên Thị Minh, Yoshihiro Nikaido, Vitaly Nikolaev, Rattima Nitisaroj, Alireza Nourian, Hanna Nurmi, Stina Ojala, Atul Kr. Ojha, Adédayọ̀ Olúòkun, Mai Omura, Emeka Onwuegbuzia, Petya Osenova, Robert Östling, Lilja Øvrelid, Şaziye Betül Özateş, Arzucan Özgür, Balkız Öztürk Başaran, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Guilherme Paulino-Passos, Angelika Peljak-Łapińska, Siyao Peng, Cenel-Augusto Perez, Natalia Perkova, Guy Perrier, Slav Petrov, Daria Petrova, Jason Phelan, Jussi Piitulainen, Tommi A Pirinen, Emily Pitler, Barbara Plank, Thierry Poibeau, Larisa Ponomareva, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Peng Qi, Andriela Rääbis, Alexandre Rademaker, Taraka Rama, Loganathan Ramasamy, Carlos Ramisch, Fam Rashel, Mohammad Sadegh Rasooli, Vinit Ravishankar, Livy Real, Petru Rebeja, Siva Reddy, Georg Rehm, Ivan Riabov, Michael Rießler, Erika Rimkutė, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Eiríkur Rögnvaldsson, Mykhailo Romanenko, Rudolf Rosa, Valentin Roșca, Davide Rovati, Olga Rudina, Jack Rueter, Kristján Rúnarsson, Shoval Sadde, Pegah Safari, Benoît Sagot, Aleksi Sahala, Shadi Saleh, Alessio Salomoni, Tanja Samardžić, Stephanie Samson, Manuela Sanguinetti, Dage Särg, Baiba Saulīte, Yanin Sawanakunanon, Kevin Scannell, Salvatore Scarlata, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Hiroyuki Shirasu, Muh Shohibussirri, Dmitry Sichinava, Einar Freyr Sigursson, Aline Silveira, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Maria Skachedubova, Aaron Smith, Isabela Soares-Bastos, Carolyn Spadine, Steinþór Steingrímsson, Antonio Stella, Milan Straka, Emmett Strickland, Jana Strnadová, Alane Suhr, Yogi Lesmana Sulestio, Umut Sulubacak, Shingo Suzuki,

Zsolt Szántó, Dima Taji, Yuta Takahashi, Fabio Tamburini, Mary Ann C. Tan, Takaaki Tanaka, Samson Tella, Isabelle Tellier, Guillaume Thomas, Liisi Torga, Marsida Toska, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Utku Türk, Francis Tyers, Sumire Uematsu, Roman Untilov, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Andrius Utka, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Aya Wakasa, Joel C. Wallenberg, Lars Wallin, Abigail Walsh, Jing Xian Wang, Jonathan North Washington, Maximilan Wendt, Paul Widmer, Seyi Williams, Mats Wirén, Christian Wittern, Tsegay Woldemariam, Tak-sum Wong, Alina Wróblewska, Mary Yako, Kayo Yamashita, Naoki Yamazaki, Chunxiao Yan, Koichi Yasuoka, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrtský, Shorouq Zahra, Amir Zeldes, Hanzhi Zhu, and Zhuravleva. 2020. Universal Dependencies 2.7. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Appendix A: Harmonisation of Amrita and UPOS tagsets

Amrita	**UPOS**	**Amrita**	**UPOS**
NN	NOUN	CVB	CCONJ
NNC	NOUN	PPO	ADP
NNP	PROPN	CNJ	CCONJ
NNPC	PROPN	DET	DET
ORD	NUM	COM	CCONJ
CRD	NUM	EMP	PART
PRP	PRON	ECH	PART
PRIN	PRON	RDW	ADP
ADJ	ADJ	QW	VERB
ADV	ADV	QM	PUNCT
VNAJ	VERB	INT	ADJ
VNAV	VERB	NNQ	NUM
VINT	VERB	QTF	NUM
VBG	NOUN	COMM	PUNCT
VF	VERB	DOT	PUNCT
VAX	VAUX		

Table 4: Harmonisation of Amrita and UPOS tagsets

Constructing a Korean Named Entity Recognition Dataset for the Financial Domain using Active Learning

Dong-Ho Jeong[a,O], **Min-Kang Heo**[a], **Hyung-Chul Kim**[b], **Sang-Won Park**[a]
DeepNatural Inc[a], Kookmin Bank[b]
dongho@deepnatural.ai, minkang@deepnatural.ai, yhdosu@kbfg.com, anson@deepnatural.ai

Abstract

The performance of deep learning models depends on the quality and quantity of data. Data construction, however, is time-consuming and costly. In addition, when expert domain data are constructed, the availability of experts is limited. In such cases, active learning can efficiently increase the performance of the learning models with minimal data construction. Although various datasets have been constructed using active learning techniques, vigorous studies on the construction of Korean data on expert domains are yet to be conducted. In this study, a corpus for named entity recognition was constructed for the financial domain using the active learning technique. The contributions of the study are as follows. (1) It was verified that the active learning technique could effectively construct the named entity recognition corpus for the financial domain, and (2) a named entity recognizer for the financial domain was developed. Data of 8,043 sentences were constructed using the proposed method, and the performance of the named entity recognizer reached 80.84%. Moreover, the proposed method reduced data construction costs by 12–25%.

1 Introduction

Rapid advancements in the field of artificial intelligence have contributed to their increased use in various fields. In the field of natural language processing, deep learning models have demonstrated a higher performance than humans in benchmark tests such as GLUE (Wang et al., 2018) and SQuAD (Rajpurkar et al., 2018). Such deep learning models require expert knowledge and time, in addition to the cost involved in computing power for model training and data construction. Therefore, it is difficult to apply deep learning models in industries readily. In particular, it is not easy to reduce the costs of learning data construction when compared to that of hardware because it requires human effort. In addition, specific fields, such as the financial domain, requires workers with expert domain knowledge and, therefore, requires more time and is more expensive. Various studies have been conducted in areas such as unsupervised learning (Taghipour and Tou Ng., 2015) and crowdsourcing (OOmen and Aroyo., 2011) to reduce the data construction costs of expert domains. The number of data constructed through such methods is larger than the number of annotations created by experts, but the quality of the constructed data is relatively low.

Active learning is a technique used by experts to construct data, with the goal of achieving optimum performance with fewer data. The active learning technique determines which data are to be learned first by interacting with users when learning data are limited. The goal of the interaction is to identify data that can be used to improve the performance of the learning model efficiently. During the early learning stages, the model is trained with a small number of learning data, and it submits queries requesting additional data that can efficiently improve its performance, thereby reducing development costs. The active learning technique is known to be effective in

Proceedings of the 17th International Conference on Natural Language Processing, pages 208–212
Patna, India, December 18 - 21, 2020.

various natural language processing fields such as information extraction, Named Entity Recognition(NER), and text classification (Settles., 2009). Various studies have been conducted on active learning for general and expert domains in other countries, but vigorous studies for constructing expert domain data are yet to be conducted in South Korea.

This study investigates active learning-based data construction for a NER system in the financial domain. The contributions of this study are as follows.

(1) It was experimentally verified that the active learning technique could effectively construct a NER corpus for the financial domain. Particularly, a cost reduction of 12.5–25% was achieved when the data of 3,000–3,500 sentences were constructed.

(2) A named entity recognizer for the financial domain with a performance of 80.84% was consequently developed.

The remainder of this paper is organized as follows. Section 2 presents a review of existing studies on active learning from Korea and other countries and an examination of the research directions in active learning. Section 3 describes the system diagram for constructing the NER corpus for the financial domain. The corpus and results are evaluated in Section 4, and the conclusions are presented in Section 5.

2 Related Research

2.1 Active Learning Technique

Passive learning represents general machine learning that learns data in sequence without a process of selecting learning data through queries. In contrast, active learning arbitrarily selects and learns data. Compared to passive learning, more learning data selection schemes exist in active learning (Settles., 2009). Uncertainty sampling is the simplest way to select the initial learning data. It is an intuitive way of assuming that if the model prediction for specific data is uncertain, information on the corresponding data is insufficient for selecting the data as the priority target data to be learned. Uncertainty sampling schemes include least confidence (LC), margin sampling, and entropy sampling according to uncertainty-measuring criteria. The most intuitive LC sampling scheme, which uses data with the lowest measured probability (e.g., softmax) first, is used in this study to develop the trained model. In other words, it first selects data that are not easily resolved by the current model as the learning data.

2.2 Research Trends in Korea and Other Countries

Studies on active learning are being conducted in various natural language processing fields, including NER (Shen et al., 2017), and in various domains (Li et al., 2012), including the financial domain (Smailović et al., 2014), in other countries (Settles., 2009). In South Korea, the active learning technique has been applied to areas such as sentence classification (Kim et al., 2012) and NER (Yoon and Oh., 2015) using person, location, and organization (PLO) representation of the named entities. However, it has not been vigorously applied to domains that require expert knowledge, such as the financial domain. In this study, active learning was applied to the construction of a NER corpus for the financial domain to recognize 40 named entities (financial institutions, broadcasting stations, economy-related institutions, and financial products such as funds and derivatives) in the finance-related domain.

3 Construction of a Named Entity Corpus for the Financial Domain using Active Learning

In this study, the NER corpus was constructed by applying active learning to the financial domain. NER identifies and classifies PLO entities in a given sentence. In the financial domain, this is an annotation task that distinguishes financial and general institutions and requires expert knowledge for tagging named entities of financial products (e.g., funds) and financial theories and phenomena. However, there are practical limitations such as the requirement for experts who perform the annotation tasks and the costs of data construction. Therefore, in this study, we investigate whether the performance of a model can be improved by minimizing the annotation tasks using active learning to handle such problems. Figure 1 presents the overall diagram of the named entity corpus construction task for the financial domain using active learning.

The task sequence is as follows. (1) First, 500 arbitrary data, which are to be annotated, are selected from the financial domain raw corpus. (2)

A named entity annotation task is performed on the corresponding data by an annotation platform. (3) The created data are then used to train the model. (4) The model is used to predict the raw corpus and select the annotation target data belonging to a specific condition. The LC scheme discussed in Section 2.1 was applied to select the annotation target data in this study. Steps (1) - (4) are repeated until the performance of the model converges. The model selects data that are expected to contribute the most to model performance by learning the sentences that are not initially predicted by the model. In other words, as data of a specific size or larger tend to contribute less to performance improvement, the model performance can be optimized with minimum data. As a result, a better performance can be obtained using our method than using random sampling, even with the construction of fewer data at a reduced cost.

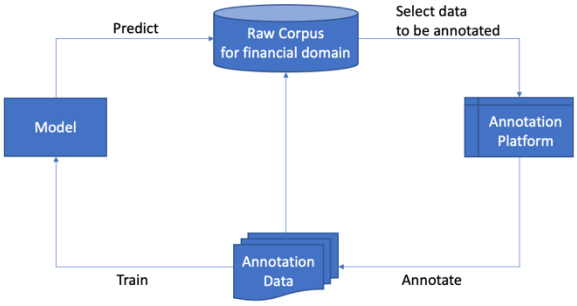


Figure 1: Schematic diagram for constructing the named entity corpus for the financial domain using active learning.

4 Experiment and Results

4.1 Data

Financial domain text was collected from the web to verify the effectiveness of the active learning model. [1] The text was collected by crawling sentences, including finance-related keywords. Of the 9,043 sentences collected, named entities in 1,043 sentences were tagged first and used as evaluation data, and the remaining 8,000 sentences were treated as the raw financial domain corpus. The tagging task was performed with 40 named entity tag sets using the annotation platform [2] shown in Figure 2.

4.2 Experiment Method

The initial annotation target data were randomly selected from the raw financial domain corpus. The selected data were tagged by the annotation platform shown in Figure 2. A morpheme analyzer provided by KB was used to separate the annotated sentences into morpheme units for use as learning data. In this study, a named entity recognizer was implemented by fine-tuning the multilingual-BERT model using the original method in (Devlin et al., 2015).

The experimental method is as follows: The model was trained using the active learning technique that samples the target data using LC and random sampling methods. Thereafter, the model was trained and evaluated by increasing the number of learning sentences by 500 to examine the performance improvement trend in the model following each iteration. The final model was generated in eight iterations, with the data increasing from 500 to 4,000 sentences. In addition, each iteration was repeated 10 times to verify the objectivity of the experiment. The hyperparameters used in the experiment are listed in Table 1.

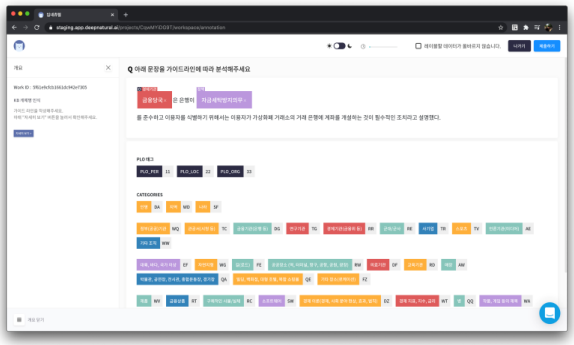


Figure 2: Processing tools of the annotation platform.

Hyperparameter	Value
Maximum sentence length	256
Batch size	6
Learning rate	3e-5
Optimizer	Adam
Epoch	10

Table 1: Font Hyperparameter information.

[1] The dataset was collected in cooperation with KB.

[2] http://app.deepnatural.ai

4.3 Experiment Results

The F1 evaluation results of the active learning model trained using LC and random sampling are presented in Figure 3. When the number of learning sentences is small (500–1,500 sentences), there are no significant differences in the model performance. It appears that at least 1,500 sentences are required to obtain meaningful information to affect the model performance. When the number of learning sentences exceeds 2,000, the model trained using the LC method performs better than that trained using random sampling. With 4,000 or more sentences, the LC and random sampling models tend to exhibit similar performances. Specifically, the F1 of random sampling is 79%, which is similar to that when the LC model is trained with 3,000–3,500 sentences. This shows that the cost can be reduced by 12.5-25% by using the active learning technique.

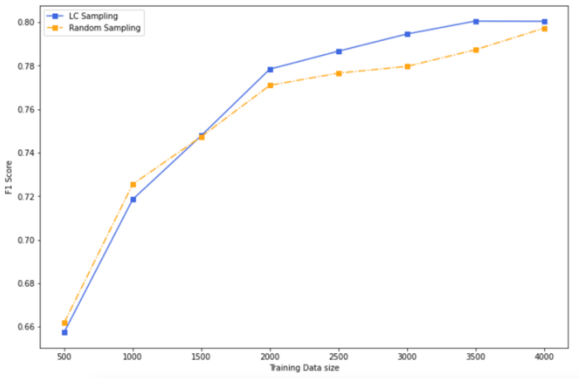


Figure 3: Model performance evaluation trend using different sampling techniques

Information on the tag set ratios of 500 sentences selected in each of the eight iterations for generating the annotation target data is presented in Table 2. Only the first four iterations are listed in Table 2 for brevity.

Step	Top 5 tag ratios (%) in LC sampling	Low 5 tag ratios (%) in random sampling
#1	AMOUNT (25)	AMOUNT (20)
	PRICE (20)	DATETIME (17)
	DATETIME (15)	PRICE (15)
	FINANCIAL_INSTITUTION (10)	FINANCIAL_INSTITUTION (15)
	PERSON (4)	PERSON (5)
#2	FINANCIAL_INSTITUTION (15)	AMOUNT (24)
	COMPANY_AND_BRAND (8)	PRICE (17)
	GOVERNMENT (7)	DATETIME (17)
	FINANCIAL_PRODUCT (7)	FINANCIAL_INSTITUTION (10)
	VALUE (6)	VALUE (6)
#3	FINANCIAL_INSTITUTION (15)	AMOUNT (23)
	DATETIME (12)	PRICE (16)
	AMOUNT (11)	DATETIME (14)
	AREA (6)	FINANCIAL_INSTITUTION (13)
	GOVERNMENT (5)	PERSON (5)
#4	PRICE (21)	AMOUNT (28)
	FINANCIAL_INSTITUTION (14)	PRICE (19)
	DATETIME (11)	DATETIME (13)
	AMOUNT (8)	FINANCIAL_INSTITUTION (9)
	FINANCIAL_PRODUCT (6)	VALUE (5)

Table 2: Top 5 tag set ratios (%) in the first four step.

The tag sets ‘AMOUNT,’ ‘PRICE,’ ‘DATETIME,’ and ‘FINANCIAL_INSTITUTION’ were mainly selected by random sampling, indicating that these tag sets frequently appear in the collected raw corpus. The tag set distribution for the entire annotated sentences also reveals that the tags ‘DATETIME’, ‘AMOUNT,’ ‘FINANCIAL_INSTITUTION’ and ‘PRICE’ appear frequently. However, in LC sampling, various tag sets appear as the upper tag set regardless of the tag set ratios in the entire dataset.

5 Conclusions

In this study, a NER corpus for the financial domain was constructed through active learning. The active learning technique could rapidly and efficiently improve the performance of the model. Tag sets that could not be easily analyzed by the model were constructed first, and it was found that more meaningful quality datasets were constructed through machine learning. From the 8,043 sentences constructed, a named entity recognizer with a F1 performance of 80.84% was developed for the financial domain. Further, it was experimentally verified that the active learning technique could provide a cost reduction of 12.5–25%.

Acknowledgments

This study was carried out as part of the 2019 Startup Growth Technology Development Project (TIPS Program, No. S2816383) supported by the Ministry of SMEs and Startups in cooperation with KB Kookmin Bank.

References

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. *arXiv preprint arXiv:1806.03822.*

Kaveh Taghipour, and Hwee Tou Ng. 2015. Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In: *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: human language technologies*. p. 314-323.

Johan Oomen, and Lora Aroyo. 2011. Crowdsourcing in the cultural heritage domain: opportunities and challenges. In: *Proceedings of the 5th International Conference on Communities and Technologies.* p. 138-149.

Burr Settles. 2009. *Active learning literature survey: Computer sciences technical report 1648*. University of Wisconsin-Madison.

Yanyao Shen, Hyokun Yun, Zachary Lipton, Yakov Kronrod, and Animashree Anandkumar. 2017. Deep active learning for named entity recognition. *arXiv preprint arXiv:1707.05928.*

Lianghao Li, Xiaoming Jin, Sinno Jialin Pan, and JianTao Sun. 2012. Multi-domain active learning for text classification. In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. p. 1086-1094.

Jasmina Smailović, Miha Grčar, Nada Lavrač, and Martin Žnidaršič. 2014. Stream-based active learning for sentiment analysis in the financial domain. *Information sciences*, 285, 181-203.

Je-wook Kim, Han-joon Kim, and Sang-goo Lee. 2012. Construction method of active learning-based learning document sets for a Bayesian document classification system. *Journal of KIISE: Software and Application*, 29(11· 12), 966-978.

Bo-hyeon Yoon and Hyo-jeong Oh. 2015. Development of a tool for semi-automatically constructing named entities in advance using active learning. *Journal of KACE*, 18(6), 81-88.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Self-Supervised Claim Identification for Automated Fact Checking

Archita Pathak
University at Buffalo (SUNY)
Buffalo, NY
architap@buffalo.edu

Mohammad Abuzar Shaikh
University at Buffalo (SUNY)
Buffalo, NY
mshaikh2@buffalo.edu

Rohini K. Srihari
University at Buffalo (SUNY)
Buffalo, NY
rohini@buffalo.edu

Abstract

We propose a novel, attention-based self-supervised approach to identify "claim-worthy" sentences in a fake news article, an important first step in automated fact-checking. We leverage *aboutness* of headline and content using attention mechanism for this task. The identified claims can be used for downstream task of claim verification for which we are releasing a benchmark dataset of manually selected compelling articles with veracity labels and associated evidence. This work goes beyond stylistic analysis to identifying content that influences reader belief. Experiments with three datasets show the strength of our model[1].

1 Introduction

The explosion of fake news on social media has resulted in global unrest and has been a major concern for governments and societies worldwide[2]. According to a recent Pew Research Study, Americans rate it as a larger problem than racism, climate change, or illegal immigration[3]. Since, it's inexpensive to create a website and easily disseminate content on the social media platforms, there is a rising need for automated fake news detection. Furthermore, AI solutions are also required to follow good practices, specifically avoiding censorship, violation of fundamental rights such as freedom of expression, and ensuring data privacy (de Cock Buning, 2018). However, to date, AI models proposed for fake news detection do not scale for detecting real-time fake news[4].

Much of the research on automated text-based fake news detection can be classified into three broad categories: (1) linguistic approach, which focuses on lexical, stylometric and pattern learning mechanisms (Potthast et al., 2017; Rashkin et al., 2017; Wang, 2017; Singhania et al., 2017; Pérez-Rosas et al., 2018); (2) network-based approach, which leverages features such as the speed and volume of propagation of fake news articles on social media platforms (Castillo et al., 2011; Yang et al., 2012; Kwon et al., 2013; Ma et al., 2015; Jin et al., 2016; Ruchansky et al., 2017; Wu and Liu, 2018); and (3) automated fact-checking approach, which is an effort to assist manual fact-checkers by automating some of their tasks such as detection and verification of claims (Graves, 2018).

While most work in automated fact-checking has been focused on claim verification task, very few methods have been proposed for detection of claims (Hassan et al., 2017; Jaradat et al., 2018; Konstantinovskiy et al., 2018). The approaches in these efforts are majorly related to *political discourse*. However, our focus is on *fake news*, which are broader than *political discourse* since (i) they are deliberately written with a divisive agenda to cause social unrest, (ii) they are not constrained to only politics, and (iii) the headline plays an equally important role in compelling people to read the article.

In this paper, we focus on articles where there is a deliberate intent to influence readers through fabricated or manipulated claims in the headline and the content. Such articles have a compelling writing style similar to the mainstream media. Hence, we build datasets containing these type of compelling articles along with veracity labels and associated evidence supporting the label of each arti-

[1] Data and code available at: https://github.com/architapathak/Self-Supervised-ClaimIdentification

[2] https://www.reuters.com/article/us-singapore-politics-fakenews-factbox/factbox-fake-news-laws-around-the-world-idUSKCN1RE0XN

[3] https://www.journalism.org/2019/06/05/many-americans-say-made-up-news-is-a-critical-problem-that-needs-to-be-fixed/

[4] https://www.technologyreview.com/s/612236/even-the-best-ai-for-spotting-fake-news-is-still-terrible/

Proceedings of the 17th International Conference on Natural Language Processing, pages 213–227
Patna, India, December 18 - 21, 2020.

cle. We, then, use these datasets to identify "claim-worthy" sentences. In our work, we define "claim" as *"statements which are important to the point of the article but one would require to have them verified."*

Our working hypothesis is that in fake news which are created to cause harm, these are the sentences most relevant to the headline. Exploiting the hypothesis that the essence of a news article is encapsulated in its headline (Jaime Sis and MERCEDES, 2009; Kuiken et al., 2017; Wahl-Jorgensen and Hanitzsch, 2009), we propose a self-supervised method to explore the *aboutness* of the content with the headline of the article to extract the most relevant sentences. Bruza and Huibers (1996) defines *aboutness* as: *an information carrier i will be said to be* ***about*** *information carrier j if the information borne by j holds in i*. The idea is taken from Information Retrieval domain where it is used to signify implications between query and document, specifically to explore the underlying meaning or concept within the document and the query (Azzopardi et al., 2009). In our work, headline is modelled as a query while each of the sentences of the article acts as a document, and we use the concept of *aboutness* to find the relevant sentences. We show that attention-based mechanisms are able to successfully capture this concept in the news article.

Contribution: In this work: (i) we introduce a self-supervised representation learning model that eliminates the prerequisite that requires human to annotate data, which is a time consuming and costly task; (ii) the proposed headline-to-sentence attention-based approach for claim identification is novel; previous unsupervised approach for this task use weak supervisory signal which does not capture the context of the article efficiently; and (iii) we propose a benchmark dataset for evidence-based fake news detection. Our dataset contains evidence for each of the fake news articles that contributes to the overall degree of veracity of the article.

2 Related Work

Claim Identification/Detection: The task of claim identification/detection was first introduced by Levy et al. (2014) who, with the help of human annotators, provided a dataset and a fundamental approach in identifying context dependent claims. In their dataset, which was originally developed by Aharoni et al. (2014), each statement indicates whether it should be considered as a context dependent claim (CDC) or not. Levy et al. (2014) reported encouraging results obtained through a supervised learning algorithm using a cascade of classifiers. A rule-based model was introduced by Eckle-Kohler et al. (2015) to bifurcate claim and premise statements in an argumentative discourse environment. However, these methods were generic to only a small set of corpora. Furthermore, Levy et al. (2017) also introduced an unsupervised approach to detect claims, which involves a weak supervisory signal "that" for training. However, this approach does not capture the *aboutness* of the article to understand the context of "claim-worthy" sentences.

In 2017, Hassan et al. (2017) introduced ClaimBuster, a platform developed by training a supervised learning model on a large annotated corpus of televised debates in the USA. Their model used SVM classifier to detect *claim-worthy factual claims* and produced a score of how important a claim is to factcheck. The 20,000 sentences in the corpus were annotated by human coders to distinguish between claim-worthy factual claims from opinions and boring statements. However, annotating a sentence as an important or unimportant claim is a non-trivial task as this decision changes depending on who's asking, political context and annotator's background (Graves, 2018).

The model proposed by Hassan et al. (2017) only learns the labelled instances and does not explore the contextual information of the written text. A context-aware approach in the political discourse environment was introduced by Gencheva et al. (2017) who created a rich representation of the sentences from 2016 US presidential debates. Their dataset was compiled by taking the outputs of factchecking of the debates from 9 factchecking organizations. Their models were created to predict if the claim would be highlighted by at least one or by a specific organization. However, the authors don't have any formal definition of claim in their paper, and their model is specific to certain organizations which led to several false positives.

Another context-aware approach for claim detection was proposed by Konstantinovskiy et al. (2018) who used sentence embeddings, pre-trained on a large dataset of NLI. This work also created a crowd-sourced annotated dataset of sentences from UK political TV shows, annotated across 7 classes. However, their classifiers for the fine-grained clas-

sification to detect 7 classes of sentences did not yield good results due to lack of enough annotated data, thus requiring more annotations which is a costly and time consuming task.

We build a model that can be trained in a self-supervised setting to overcome the challenges associated with annotated dataset of claims. We also use attention-based approach to capture *aboutness* and rich contextual information between headline and all the sentences of the article. The performance on manually created test sets demonstrate promising results in identifying "claim-worthy" sentences even when no sentence-level annotation was used for training.

Fake News Dataset: A variety of fake news datasets have been released in the recent years, most notably Buzzfeed[5] and Stanford (Allcott and Gentzkow, 2017) datasets containing list of popular fake news articles from 2016 US presidential elections. However, these datasets only contain webpage URLs of the original article and majority of them don't exist anymore. Following this, several other datasets were published such as Fake news challenge dataset[6] which was used for the task of stance detection; *Getting Real about Fake News* Kaggle dataset[7] which was created by using BS detector tool; and FakeNewsCorpus[8] which is an open-source large scale collection of fake news articles. However, these articles are labelled as fake based on the domain of the websites they come from. Since, the content of these articles are not verified for degree of veracity, using them directly for training may lead to several false positives.

This problem was overcome by recently released large dataset, NELA-GT-2018 (Norregaard et al., 2019), which contains articles with ground truth ratings retrieved from 8 different assessment sites. However, the label definitions are not generic and dependent on the external organizations. Pathak and Srihari (2019) also introduced intuitive ground truth labels based on the degree of veracity of the fake news articles, however, the dataset is not publicly available. Additionally, they also do not specify the relationship of their labels with the labels used by established fact-checking organizations. Furthermore, due to lack of evidence in these datasets, they cannot be used for downstream task of evidence-based verification, which is one of the motivations of this paper. We overcome all these limitations in our datasets described in the following section.

3 Datasets

We introduce two datasets of compelling fake news articles which have writing style similar to mainstream media. The first dataset, DNF-700, where DNF stands for ***DisiNFormation***, contains articles on politics published within 4 months of 2016 US Presidential Elections from questionable sources (non-mainstream). To compile this dataset, we first extracted fake news articles from working web page URLs of Stanford dataset (Allcott and Gentzkow, 2017). However, majority of webpage URLs in this dataset are expired and we could extract only 26 fake news articles. Therefore, we then used "*Getting real about fake news*" Kaggle[8] dataset to sample more articles on politics. Since, most of the articles in this dataset contain anomalies (eg: incomplete article, social media comments labelled as fake etc.), we manually verified the writing style and discarded obvious fakes - articles with poor grammar and excessive usage of punctuations. However, the degree of veracity of each article in this dataset is not checked and some articles may contain personal opinions.

The second dataset, DNF-300, is more sophisticated subset of DNF-700, containing 290 compelling articles on Politics and 10 on Health/Medical news. Unlike other fake news datasets in which veracity and evidence for articles are not provided, DNF-300 contains articles associated with veracity labels as well as corresponding evidence. The process of annotating this dataset involves identifying sentences from each article based on their persuasive tone and relevance with the headline. These sentences were then queried on the web and top 10 results were considered to gather evidence from credible sources[9]. Based on the evidence found, we label the entire article into four categories: {(0) *false*; (1) *partial truth*; (2) *opinions stated as fact*; (3) *true*}. These labels are inspired by (Pathak and Srihari, 2019); Table-1 shows the description and distribution of these labels while the comparison with two popular fact-checking websites is displayed in Figure-1. An

[5]https://www.buzzfeednews.com/article/craigsilverman/these-are-50-of-the-biggest-fake-news-hits-on-facebook-in

[6]http://www.fakenewschallenge.org/

[7]https://www.kaggle.com/mrisdal/fake-news

[8]https://github.com/several27/FakeNewsCorpus

[9]Credible sources were extracted from https://mediabiasfactcheck.com/ The sources range between left, center and right biased news sources

Label	False	Partial True	Opinion Stated As Fact	True
Desc.	(i) No evidence could be found, or (ii) found evidence to refute the entire article	Article about true event, however, found evidence refuting some of the claims	Article contain false/manipulated claims, however, it's an opinion article which cannot be labelled as fake	Found evidence supporting the entire article
Total	126	75	79	20

Table 1: DNF-300 label description and distribution. Claims here are the sentences manually selected based on their persuasive tone and relevance with the headline. Interestingly, some of the articles, which were labelled as fake in other datasets due to the domain of publishing website, turned out to be true news.

example from the dataset is shown in Table-2

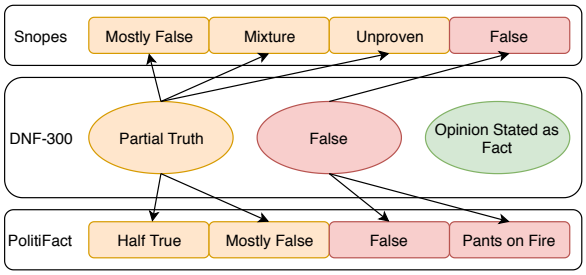


Figure 1: Label Comparison with Snopes and PolitiFact ratings.

This dataset is also a key contribution of this paper as the articles are manually read and verified. Additionally, the dataset contains two novel features which are essential for the fake news verification task: (i) generic veracity-based label set, independent of any external organization, and (ii) ground truth evidence corresponding to each label.

In addition to these two datasets, we also train our model for claim identification on the dataset introduced for context dependent claim detection (CDCD) by Levy et al. (2014). Although this dataset (CDC) does not contain fake news articles, it has manually annotated sentences based on their relevance to a certain topic. These annotations were utilized for the evaluation of our self-supervised learning model described in the following section. More details on the datasets and examples can be found in Appendix.

4 Architecture

Problem Definition: Given an article with a set of sentences $S = \{S_1, S_2, ...S_i, ...S_n\}$ and a headline H, the task of our multihead attention claim identification network (MA-CIN) is to extract the sentence most relevant to the headline. Our self-supervised model exploits the rich contextual information to extract the relevant sentences which are considered as "claim-worthy".

Approach: For this task, we implement two types of attention: (i) self-attention on all sentence vectors so that each sentence S_i is aware of all other sentences in S; (ii) cross-attention of headline vector on each sentence vector, so that all self-attended sentences are also aware of the headline's context. We then generate headline based on the context-aware sentences, and compare it with the original headline in three different settings as listed below:

1. **Headline Vector (MA-CIN (HV)):** In this setting, the original headline vector acts as the supervisory signal for self-supervised learning. We minimize the mean squared error (MSE) between the generated and the original headline vectors for training.
2. **Headline One-Hot Word Vector (MA-CIN (OHWV)):** In this setting, the words in the original headline act as the supervisory signal. We use LSTM (Hochreiter and Schmidhuber, 1997) to predict at most 50 words, from a vocabulary of 20,000 words, to generate a one-hot-vector for each word of the new headline. We then minimize the categorical cross-entropy error (CCE) at each time step corresponding to each word in original and new headlines for training.
3. **Combined HV & OHWV (MA-CIN (Combined)):** In this setting, both original headline vector and the words act as supervisory signal. Therefore, we combine the two loss functions mentioned above to train the model.

For this, we build several layers in our architecture (see Figure-2), which are delineated as follows:

4.1 Sentence Embeddings

Each sentence S_i in S, and headline H are converted to a fixed length 300-dimensional vector, s_i and h, such that $s_i, h \in R^{1\times D}$, where $D = 300$. For uniformity, we calculate the maximum number of sentences L that an article can contain in the respective corpus. Next, we zero pad the difference in the quantity of sentence vectors in each article such that every article can be represented as a vector $A \in R^{L\times D}$.

Headline: Allergens in Vaccines Are Causing Life-Threatening Food Allergies
It would probably surprise few people to hear that food allergies are increasingly common in U.S. children and around the world . According to one public health website , food allergies in children aged 0-17 in the U.S. increased by 50% from 1997 to 2011. Although food allergies are now so widespread as to have become almost normalized, it is important to realize that millions of American children and adults suffer from severe rapid-onset allergic reactions that can be life-threatening. Foods represent the most common cause of anaphylaxis among children and adolescents. The United Kingdom has witnessed a 700% increase in hospital admissions for anaphylaxis and a 500% increase in admissions for food allergy since 1990. The question that few are asking is why life-threatening food allergies have become so alarmingly pervasive. A 2015 open access case report by Vinu Arumugham in the Journal of Developing Drugs , entitled " Evidence that Food Proteins in Vaccines Cause the Development of Food Allergies and Its Implications for Vaccine Policy ," persuasively argues that allergens in vaccinesand specifically food proteinsmay be the elephant in the room. As Arumugham points out, scientists have known for over 100 years that injecting proteins into humans or animals causes immune system sensitization to those proteins. And, since the 1940s, researchers have confirmed that food proteins in vaccines can induce allergy in vaccine recipients. Arumugham is not the first to bring the vaccine-allergy link to the publics attention. Heather Fraser makes a powerful case for the role of vaccines in precipitating peanut allergies in her 2011 book, The Peanut Allergy Epidemic: Whats Causing It and How to Stop It.
Type: 1 (*Partial Truth*) **Authors**:Admin - Orissa **URLs**: galacticconnection.com
Evidence: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3890451/
Reason: The key claim is written in such a way so that it misleads people in thinking all the food related allergies in US are caused by vaccines. Found evidence which says these type of allergies are rare.

Table 2: An example on *Partial Truth* type from DNF-300 dataset.

4.2 1D Convolution

To effectively capture local relevance, we leverage 1D-CNN (LeCun et al., 1998) to extract the features from the article vector A. For our experiments the kernel size for each convolution layer is $K \times D \times C$, where K is kernel-width and C is the number of filters. This means the network will process K sentences at a time. The size of K and C is a hyper-parameter and as per our experiments, we set $K = 4$ with an assumption that not more than 4 consecutive sentences will be relevant to each other.

4.3 Self Attention

Inspired by the attention implementation in (Zhang et al., 2018; Vaswani et al., 2017), to capture global relevance, the article features from the previous 1D-CNN layer are transformed into feature spaces q, k to calculate attention, where $q(x) = W_q x$ and $k(x) = W_k x$.

$$\beta_{j,i} = \frac{exp(z_{ij})}{\sum_{i=1}^{N} exp(z_{ij})}, \text{where } z_{ij} = q(x_i)^T k(x_j) \tag{1}$$

$\beta \in R^{N \times N}$ is the attention coefficient, which is the normalized relevance score between the sentence x_i and x_j. β is then matrix multiplied by v, where $v(x) = W_v x$, to obtain the context rich output $o_j \in R^{C \times 1}$.

$$o_j = \sum_{i=i}^{N} \beta_{j,i} v(x_i), \text{where } o_j \in \{o_1, o_2, \ldots, o_N\} \tag{2}$$

Finally, the output of the self-attention layer is $o \in R^{C \times N}$, which is computed as

$$o_j = g(o_j), \text{where, } g(x) = W_g x \tag{3}$$

In the above equations, $x \in R^{C \times N}$ is obtained after applying 1D convolution on sentence vectors, $W_q \in R^{\overline{C} \times C}$, $W_k \in R^{\overline{C} \times C}$, $W_v \in R^{\overline{C} \times C}$, $W_g \in$

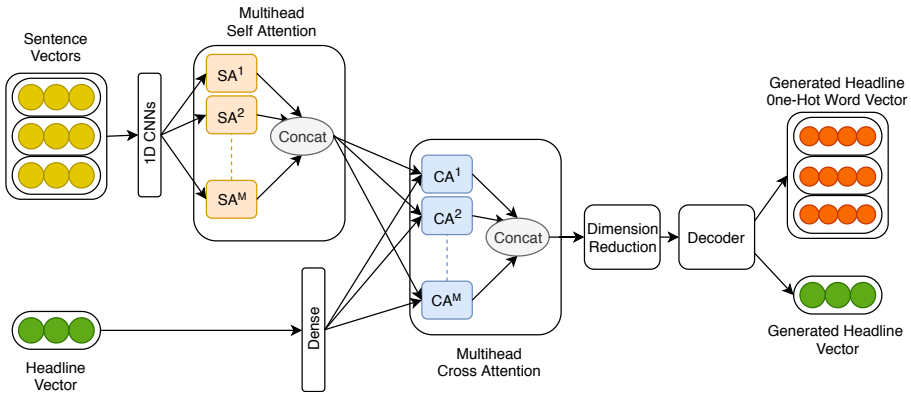


Figure 2: Architecture of Multihead Attention - Claim Identification Network (MA-CIN). The model is trained by using self-supervised learning approach using three variants of supervisory-signal - headline vector, headline words and the combination of both vector and words.

$R^{C\times\overline{C}}$ and output $o \in R^{C\times N}$. Following the work by (Zhang et al., 2018) we preferred the value of $\overline{C} = \frac{C}{8}$ for computation effectiveness. We also multiply a λ and γ, learnable scale parameters, to the output of our attention module and input vector respectively to allow the network to choose between local and global sentences effectively.

$$o = \gamma x + \lambda o \tag{4}$$

γ is initialized to 1 and λ is initialized to 0, so as to allow the local context to be captured effectively during the early iterations and as the value of λ increases it allows the network to add more context to the representation.

4.4 Multihead Concatenation

In the architecture, we could apply self attention to the input x M times resulting into M attention heads. The output of one attention head is denoted by o. We concatenate the outputs o^M to get a richer representation allowing the network to capture various relationships.

$$msa\_o = \Big\|_{i=1}^{M} o^i = o^1 \| \ldots \| o^M \tag{5}$$

where, $msa\_o \in R^{MC\times N}$ is the long range context aware output of multihead self attention. Here, $\|$ denotes concatenation across axis C.

4.5 Cross Attention

The headline vector is transformed into a feature space $\overline{h} = W_h h$, where $\overline{h} \in R^{\overline{C}\times 1}$ and then, it's relevance is calculated with msa_o, obtained from the previous layer, by using equations defined in 4.3. Finally, after applying multihead concatenation using 5, we obtain headline-context aware representation, $mca\_o \in R^{MC\times N}$. We fix $M = 4$ for all our experiments.

4.6 Loss Function

To generate the headline vector d_h as close to the input headline vector h, we apply Mean Squared Error between d_h and h and calculate the headline vector generation loss L_v

$$L_v = \frac{1}{n}\Big(\sum_{i=1}^{n}(d_{h_i} - h_i)^2\Big) \tag{6}$$

For estimating the probability of a word from the vocab in the predicted headline we calculate the cross-entropy between the predicted headline words d_{hw} and input headline one-hot vector HW.

$$L_w = -\sum_i d_{hw_i} \log(HW_i) \tag{7}$$

The total loss $L_{total} = L_v + L_w$ is then evaluated for all samples $b \in B$, where B is one batch.

5 Experiments and Evaluation

5.1 Training Setup

We train our Multihead Attention model for Claim Identification, MA-CIN, on datasets mentioned in Section 3. The CDC dataset contains total of 522 articles. Amongst these, there are 47 articles with 8 or more annotated claim sentences which are considered as evaluation set (CDC_Eval) for this dataset. Next, for DNF-300 and DNF-700, we asked two annotators to manually tag at least 5 sentences as "claim-worthy" in each of the 50 articles. Sentences which were consented by both

Dataset	Configuration	CDC_Eval			DNF_Eval		
		Prec.	Rec.	F1	Prec.	Rec.	F1
$Spacy$	Baseline	0.09	0.14	0.11	0.33	0.42	0.37
CDC	Baseline (Levy et al., 2014)	0.23	-	-	-	-	-
	MA-CIN(HV)	0.18	0.08	0.11	0.39	0.53	0.45
	MA-CIN(OHWV)	0.25	0.10	0.15	0.40	0.54	0.46
	MA-CIN(Combined)	**0.26**	**0.11**	**0.16**	**0.42**	**0.57**	**0.48**
$DNF-700$	MA-CIN(HV)	0.20	0.09	0.12	0.37	0.54	0.44
	MA-CIN(OHWV)	0.19	0.08	0.11	0.40	0.5	0.44
	MA-CIN(Combined)	**0.28**	**0.12**	**0.16**	**0.41**	**0.55**	**0.47**
$DNF-300$	MA-CIN(HV)	0.19	0.08	0.11	0.39	0.53	0.48
	MA-CIN(OHWV)	**0.25**	**0.11**	**0.15**	0.38	0.53	0.45
	MA-CIN(Combined)	0.24	0.10	0.14	**0.42**	**0.57**	**0.48**

Table 3: Comparison of MA-CIN model configurations over three datasets and two evaluation sets for identification of "claim-worthy" sentences.

	Headline: Clinton Received Debate Questions Week Before Debate			
0	The first presidential debate was held and Hillary Clinton was proclaimed the winner by the media.	-	-	0.41
1	Indeed Clinton was able to turn in a strong debate performance, but did she do so fairly?	-	-	0
2	Multiple reports and leaked information from inside the Clinton camp claim that the Clinton campaign was given the entire set of debate questions an entire week before the actual debate.	GT	PD	1
3	Earlier last week an NBC intern was seen hand delivering a package to Clinton's campaign headquarters, according to sources.	-	PD	0.73
4	The package was not given to secretarial staff, as would normally happen, but the intern was instead ushered into the personal office of Clinton campaign manager Robert Mook	-	PD	0.68
5	Members of the Clinton press corps from several media organizations were in attendance at the time, and a reporter from Fox News recognized the intern, but said he was initially confused because the NBC intern was dressed like a Fed Ex employee.	-	-	0.46
6	The reporter from Fox questioned campaign staff about the intern, but campaign staff at first claimed ignorance and then claimed that it was just a Fed Ex employee who had already left.	-	-	0.67
7	No reporters present who had seen the intern dressed as a Fed Ex employee go into Mook's office saw him leave by the same front entrance.	-	-	0.49
8	The Fox reporter who recognized the intern also immediately looked outside of the campaign headquarters and noted that there were no Fed Ex vehicles parked outside.	-	-	0.51
9	Clinton seemed to have scripted responses ready for every question she was asked at the first debate.	GT	-	0.37
10	She had facts and numbers memorized for specific questions that it is very doubtful she would have had without being furnished the questions beforehand.	GT	-	0.63
11	The entire mainstream media has specifically been trying to portray Trump as a racist and a poor candidate.	-	-	0.24
12	By furnishing Clinton with the debate questions NBC certainly hoped to make Clinton appear much more knowledgeable and competent than Trump.	GT	PD	0.79
13	And though it is unlikely that anyone will be able to conclusively prove that Clinton was given the debate questions, it seems both logical and likely.	GT	PD	0.7

Figure 3: Interpretation of relevance of sentences with the headline of an example article from DNF-300. GT and PD indicate ground truth and top-5 predicted "claim-worthy" sentences, respectively. MA-CIN model was able to predict 3 most relevant sentences correctly. Last column shows the attention weights between headline and each of the sentences of the article. Sentence 2 has been correctly predicted as the most relevant while sentence 1 is the least relevant.

the annotators as "claim-worthy" were finalized as ground truth claims for these 50 articles, and used as testing set for evaluating the model performance on DNF datasets. The remaining 475 articles from CDC, 250 articles from DNF-300, and 650 articles from DNF-700 were split into 5 folds to train the model using a 5-Fold cross validation (Kohavi, 1995), where we use 4 folds for training and 1 fold for validation. Each of the three settings, described in Section- 4: MA-CIN(HV), MA-CIN(OHWV) and MA-CIN(Combined), was trained with each of the three datasets, and evaluated on DNF_Eval and CDC_Eval. Total number of parameters for these three settings are 15,012,916 (10,240 non-trainable), 40,975,656 (10,240 non-trainable) and 41,812,564 (12,288 non-trainable) respectively. All other network parameters are displayed in supplemental material.

In each setting, we use batch normalization, ReLU non-linearity as an activation function, and a dropout of 0.5 for every convolution operation. We trained all the models for 2000 epochs, where, for every training we used Adam optimizer with a learning rate $lr = 0.0001$, $\beta_1 = 0.99$ and $\beta_2 = 0.0$. There was no weight decay set as the model was trained in a self-supervised setting with finite epochs and an already small learning rate. Glove 300D word embedding was used for all our experiments and the number of input sentences was set to 500. The models were trained on three 11GiB Nvidia 1080Ti GPUs in parallel.

5.2 Evaluation Metrics

We evaluate MA-CIN models on two evaluation sets, DNF_Eval and CDC_Eval. With self-

supervised setting we first rank the sentences based on relevance with the headline and then extract the top five sentences along with their sentence ids as "claim-worthy" sentences. For evaluation on DNF_Eval, we calculate the *true positives*(TP), *false positives*(FP) and *false negatives*(FN) from ground truth claim ids. To evaluate on CDC_Eval, since we do not have ground truth claim ids, we calculate cosine similarity between the extracted sentences and the ground truth claims. We experiment with various similarity threshold to calculate TP, FP and FN, and set the final threshold to 0.95 to report best performing results. Finally, these metrics are used to report Precision@1, Recall@1 and F-1 scores.

5.3 Results

Table-3 shows the performance of baseline (CDC) (Levy et al., 2014) and three variants of MA-CIN models. We report two baselines - (1) spacy, and (2) Levy et al. (2014) using supervised learning method on CDC dataset which contains annotated claims. Since, Levy et al. (2014) do not report Recall and F1 scores, we have reported their Precision@1 score in this paper. We also train MA-CIN models on this dataset by removing all the annotations for self-supervised training. We observe that:

1. The combined variant of our self-supervised approach performs slightly better than the baseline on the CDC dataset. This shows that, MA-CIN models are able to learn similar properties as the baseline but without any sentence-level annotations. Thus, this eliminates the need to have an annotated dataset for claim identification.
2. MA-CIN models give comparable results on all three datasets. This shows the scalability of the models to identify "claim-worthy" sentences from any given article.
3. The combined variant of MA-CIN, which generates both the headline vector and the word in headline, performs better on all the datasets, except one: MA-CIN (OHWV) model trained on DNF-300 and evaluated on CDC_Eval performs slightly better than the combined model, however, the difference in the performance is very small.

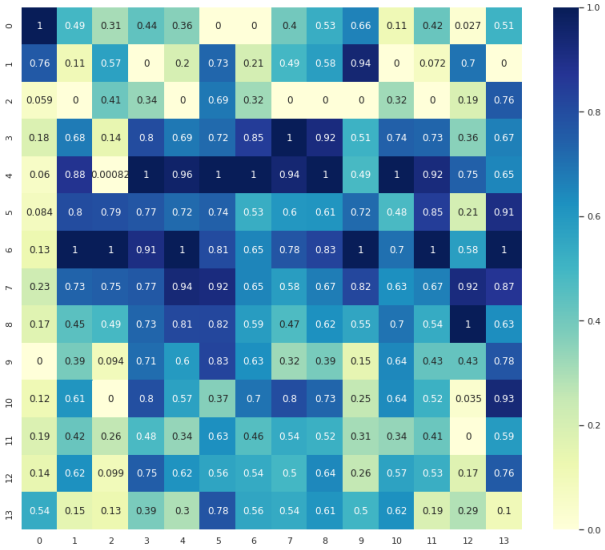


Figure 4: Interpretation of sentence-to-sentence relevance through attention weights.

6 Discussion

6.1 Analyzing Attention Weights

Attention weights help make the model interpretable to the end users by depicting relationship between all sentences as well as with the headline. From Figure-3, we can see that out of the top-5 predicted claims, 3 of them are present in the human evaluated test set. The last column, which contains attention coefficients between the headline and each sentence, depicts some interesting results -

(i) based on the human evaluation, the sentence having the least relevance with the headline is sentence 1. While this sentence contains words also present in the headline, the underlying meaning is not the same. This has been successfully captured by MA-CIN model by predicting sentence 1 as the least relevant claim;

(ii) further, highly ranked sentences 2, 12, and 13 have been correctly predicted as relevant claims by the model. This shows the model's ability in learning the semantic relationship between the headline and the content of the article, and subsequently putting importance on sentences that are relevant to the headline's underlying meaning. This property, which is also called "*aboutness*", is efficiently exhibited by the model.

(iii) sentence 3, which is predicted by MA-CIN model as relevant with a score of 0.73, is not present in the ground truth. This indicates that the two annotators did not agree to have this sentence verified, even if it is relevant to the point of the

article. To analyze this further, we plan to conduct user studies as one of the future avenues.

(iv) sentence 4 is also predicted as a relevant claim but it's missing from the ground truth since the annotators did not agree to have this verified. The reason for this prediction could be because self-attention is able to identify the premise of highly relevant sentences. Hence, sentence 4, which is the continuation of highly relevant sentence 3, is also given importance by the headline. This relevance between sentences 3 and 4 is depicted in Figure-4, where the attention weight between these two is the highest.

From Figure-4, we also observe that:

(i) sentence 4 is highly relevant to sentences 3 to 8, which is intuitive, since the story of the intern forms the premise of the claims in the article;

(ii) sentences 2 and 4 have been shown to have the least relevance with each other which is also true as shown in Figure-3. The two sentences, if considered in isolation, make two different claims which are not related to each other;

(iii) the model has made sure that a sentence does not assign high relevance to itself as it would be counter-intuitive.

6.2 Limitations

Since, the evaluation methodologies for CDC dataset has not been explained clearly, in our paper, we have considered vector cosine similarity between the ground truth claim in the CDC_Eval and the extracted claim from the model which may leave a margin of error in the evaluation scores. Additionally, ground truth in DNF_Eval is manually generated and may contain subjective biases. Although these biases have been overcome by MACIN models, as explained in 6.1, but we also plan to enhance the ground truth judgement using crowdsourced annotation. We intend to use these annotations to fine-tune the models.

7 Conclusion and Future Work

In this work, we build a novel, self-supervised approach to identify "claim-worthy" sentences - an important task for automated fact checking. The focus of this work is on fake news articles where there is a deliberate intent to influence people or cause social unrest. We have introduced novel datasets of such articles with features essential for the downstream task of fake news verification. Using powerful attention models, we explore the notion of *aboutness* of the headline and the content of the article to identify "claim-worthy" sentences. Experiments with three datasets show the strength of our model architecture in overcoming human-induced biases, which is quite common when using sentence-level claim-annotated datasets. Based on the comparison with the baseline, which was implemented using annotated dataset, we show that our models do not require annotated claims for training to identify claim-worthy sentences efficiently. We have also showed that our model is scalable to any dataset with topic and content.

Future work involves increasing the robustness of the models presented in this paper. We plan to use crowdsourced annotation on the dataset released with this paper to measure the *influence* of the article on general readers and then use these indicators to fine-tune our models. Experimentation with more robust sentence encoders is another avenue of future work. Additionally, going forward, we plan to identify a maximum of 3 claims per article which will be used for evidence-based fake news detection. We also plan to expand the dataset, presented in this work, to include fake news articles on topics other than Politics and Health.

References

Ehud Aharoni, Anatoly Polnarov, Tamar Lavee, Daniel Hershcovich, Ran Levy, Ruty Rinott, Dan Gutfreund, and Noam Slonim. 2014. A Benchmark Dataset for Automatic Detection of Claims and Evidence in the Context of Controversial Topics. In *Proceedings of the First Workshop on Argumentation Mining*, pages 64–68, Baltimore, Maryland. Association for Computational Linguistics.

Hunt Allcott and Matthew Gentzkow. 2017. Social media and fake news in the 2016 election. *Journal of economic perspectives*, 31(2):211–36.

Leif Azzopardi, Gabriella Kazai, Stephen Robertson, Stefan Rüger, Milad Shokouhi, Dawei Song, and Emine Yilmaz. 2009. *Advances in Information Retrieval Theory: Second International Conference on the Theory of Information Retrieval, ICTIR 2009 Cambridge, UK, September 10-12, 2009 Proceedings*, volume 5766. Springer.

PD Bruza and Theo WC Huibers. 1996. A study of aboutness in information retrieval. *Artificial Intelligence Review*, 10(5-6):381–407.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 675–684. ACM.

M de Cock Buning. 2018. *A multi-dimensional approach to disinformation: Report of the independent high level group on fake news and online disinformation*. Publications Office of the European Union.

Judith Eckle-Kohler, Roland Kluge, and Iryna Gurevych. 2015. On the Role of Discourse Markers for Discriminating Claims and Premises in Argumentative Discourse. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2236–2242, Lisbon, Portugal. Association for Computational Linguistics.

Pepa Gencheva, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, and Ivan Koychev. 2017. A context-aware approach for detecting worth-checking claims in political debates. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 267–276, Varna, Bulgaria. INCOMA Ltd.

D Graves. 2018. Understanding the promise and limits of automated fact-checking.

Naeemul Hassan, Fatma Arslan, Chengkai Li, and Mark Tremayne. 2017. Toward automated fact-checking: Detecting check-worthy factual claims by claimbuster. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1803–1812. ACM.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Mercedes Jaime Sis and MERCEDES. 2009. Titles or headlines? anticipating conclusions in biomedical research article titles as a persuasive journalistic strategy to attract busy readers. *Miscelnea: A Journal of English and American Studies*, 39:29–51.

Israa Jaradat, Pepa Gencheva, Alberto Barrón-Cedeño, Lluís Màrquez, and Preslav Nakov. 2018. ClaimRank: Detecting check-worthy claims in Arabic and English. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 26–30, New Orleans, Louisiana. Association for Computational Linguistics.

Zhiwei Jin, Juan Cao, Yongdong Zhang, and Jiebo Luo. 2016. News verification by exploiting conflicting social viewpoints in microblogs. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Ron Kohavi. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Lev Konstantinovskiy, Oliver Price, Mevan Babakar, and Arkaitz Zubiaga. 2018. Towards automated factchecking: Developing an annotation schema and benchmark for consistent automated claim detection. *arXiv preprint arXiv:1809.08193*.

Jeffrey Kuiken, Anne Schuth, Martijn Spitters, and Maarten Marx. 2017. Effective headlines of newspaper articles in a digital environment.

Sejeong Kwon, Meeyoung Cha, Kyomin Jung, Wei Chen, and Yajun Wang. 2013. Prominent features of rumor propagation in online social media. In *2013 IEEE 13th International Conference on Data Mining*, pages 1103–1108. IEEE.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2323.

Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. 2014. Context Dependent Claim Detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1489–1500, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Ran Levy, Shai Gretz, Benjamin Sznajder, Shay Hummel, Ranit Aharonov, and Noam Slonim. 2017. Unsupervised corpus-wide claim detection. In *ArgMining@EMNLP*.

Jing Ma, Wei Gao, Zhongyu Wei, Yueming Lu, and Kam-Fai Wong. 2015. Detect rumors using time series of social context information on microblogging websites. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1751–1754, New York, NY, USA. ACM.

Jeppe Norregaard, Benjamin D. Horne, and Sibel Adali. 2019. NELA-GT-2018: A large multi-labelled news dataset for the study of misinformation in news articles. *CoRR*, abs/1904.01546.

Archita Pathak and Rohini Srihari. 2019. BREAKING! presenting fake news corpus for automated fact checking. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 357–362, Florence, Italy. Association for Computational Linguistics.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2018. Automatic detection of fake news. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3391–3401, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2017. A stylometric inquiry into hyperpartisan and fake news. *arXiv preprint arXiv:1702.05638*.

Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2931–2937.

Natali Ruchansky, Sungyong Seo, and Yan Liu. 2017. Csi: A hybrid deep model for fake news detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 797–806, New York, NY, USA. ACM.

Sneha Singhania, Nigel Fernandez, and Shrisha Rao. 2017. 3han: A deep neural network for fake news detection. In *International Conference on Neural Information Processing*, pages 572–581. Springer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *arXiv:1706.03762 [cs]*. ArXiv: 1706.03762.

Karin Wahl-Jorgensen and Thomas Hanitzsch. 2009. The Handbook Of Journalism Studies. Technical report.

William Yang Wang. 2017. " liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*.

Liang Wu and Huan Liu. 2018. Tracing fake-news footprints: Characterizing social media messages by how they propagate. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 637–645, New York, NY, USA. ACM.

Fan Yang, Yang Liu, Xiaohui Yu, and Min Yang. 2012. Automatic detection of rumor on sina weibo. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, page 13. ACM.

Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. 2018. Self-attention generative adversarial networks.

A Appendices

A.1 Definitions

Fake News: Articles where there is a deliberate intent to influence readers through fabricated or manipulated claims in the headline and the content. Such articles have a compelling writing style similar to the mainstream media.

"Claim-worthy": Statements which are important to the point of the article but one would require to have them verified.

Compelling Fake News Articles: Articles which make persuasive claims in headline and content, that may influence readers to believe a fabricated/manipulated story.

Credible Sources: Mainstream media, established fact-checking websites and Government documents.

Questionable Sources: Non-mainstream media like infowars, naturalnews, breitbart etc.

A.2 Experiment Architectures

A.2.1 Vector Generation

Architecture setting for generating Headline Vector (HV) displayed in Figure-5

input	Model	Output size	[Kernel size, Filters, Strides], Repeats
sentence vectors		500x300	
	conv1d_1	500 x 256	[4, 256, 1] x 1
	conv1d_2	500 x 512	[4, 256, 1] x 1
	SelfAttention	500 x 512	[1, 64, 1] x 4
	Concat	500 x 2048	
	conv1d_3	500 x 512	[4, 256, 1] x 1
headline vector		1 x 300	
	conv1d_4	1 x 512	[1, 512, 1] x 1
	CrossAttention	500 x 512	[1, 64, 1] x 4
	Concat	500 x 2048	
	conv1d_5	250 x 512	[4, 512, 2] x 1
	conv1d_6	125 x 512	[4, 512, 2] x 1
	conv1d_7	63 x 512	[4, 512, 2] x 1
	conv1d_8	32 x 512	[4, 512, 2] x 1
	Global Pooling	512	
	FC_1	1024	
	output_vector	300	

Figure 5: Architecture setting for generating Headline Vector(HV).

A.2.2 Word Generation

Architecture setting for generating Headline Vector Word Probabilities (OHWV) displayed in Figure-6

A.3 DNF-700 Dataset Details

Each article is identified by an *id*. The content of the article is stored in a separate text files having file name "article_*id*", for example, article_122. A JSON file is also provided with the following fields:

id: Unique identifier of the article starting from 0.
authors: Authors of the article.
headline: Headline of the article.
type: "fake" (articles from Stanford and Buzzfeed datasets which are already proven fake); and "questionable" (articles from *Getting Real About Fake News* Kaggle dataset which require manual

input	Model	Output size	[Kernel size, Filters, Strides], Repeats
sentence vectors		500x300	
	conv1d_1	500 x 256	[4, 256, 1] x 1
	conv1d_2	500 x 512	[4, 256, 1] x 1
	SelfAttention	500 x 512	[1, 64, 1] x 4
	Concat	500 x 2048	
	conv1d_3	500 x 512	[4, 256, 1] x 1
headline vector		1 x 300	
	conv1d_4	1 x 512	[1, 512, 1] x 1
	CrossAttention	500 x 512	[1, 64, 1] x 4
	Concat	500 x 2048	
	conv1d_5	250 x 512	[4, 512, 2] x 1
	conv1d_6	125 x 512	[4, 512, 2] x 1
	conv1d_7	63 x 512	[4, 512, 2] x 1
	conv1d_8	32 x 512	[4, 512, 2] x 1
	Global Pooling	512	
	Repeat	50 x 512	
	Bi-LSTM	50 x 1024	
	TimeDistributed Dense, softmax	50 x 20000	

Figure 6: Architecture setting for generating Headline Vector Word Probabilities (OHWV).

verification of the degree of veracity)
urls: Source/domain URL of the article.

A.4 DNF-300 Dataset Details

DNF-300 is more sophisticated subset of DNF-700 with additional fields based on manual verification of the article. The JSON file of this dataset contains following fields:

id: Unique identifier of the article starting from 0.
authors: Authors of the article.
headline: Headline of the article.
type: {(0) False; (1) Partial Truth; (2) Opinions Stated As Fact; (3) True}
urls: Source/domain URL of the article.
evidence: URL of credible sources supporting or refuting the article. This field is empty when no evidence were found which talked about the claims made in this article. This means, the claims are innovated lies. In such cases, the *type* field is set as 0.
reason: Reason about the verdict. It can be one of the following:

1. Based on Snopes rating 'False' which means 'the primary elements of a claim are demonstrably false.'
2. Based on Snopes rating 'Unproven' which means 'insufficient evidence exists to establish the given claim as true, but the claim cannot be definitively proved false.'
3. Based on Snopes rating 'Mixture' which means 'a claim has significant elements of both truth and falsity to it such that it could not fairly be described by any other rating.'
4. Based on Snopes rating 'Mostly False' which means 'the primary elements of a claim are demonstrably false, but some of the ancillary details surrounding the claim may be accurate.'
5. The key claim is false (based on Snopes rating), however, the article also contains opinions stated as fact.
6. Snopes mentiones that a true story was manipulated to mislead people.
7. The key claims are true but exaggerated by adding personal opinions stated as fact.
8. No reports from trusted sources for the key claims.
9. True story manipulated to mislead readers by making unverifiable claims such as *'some_claim'*.
10. Article is fraught with opinions stated as fact about a true event.
11. Found evidence to refute key claims.
12. Article contains opinions stated as fact.
13. Evidence found to support key claims.

Topic: The sale of violent video games to minors should be banned		
S1	Violent video games can increase children's aggression	V
S2	Video game addiction is excessive or compulsive use of computer and video games that interferes with daily life	X
S3	Many TV programmers argue that their shows just mirror the violence that goes on in the real world	X
S4	Violent video games should not be sold to children	X
S5	Video game publishers unethically train children in the use of weapons	V

Figure 7: : Example for CDCs and for statements that should not be considered as CDCs. The V and X indicate if the candidate is a CDC for the given Topic, or not, respectively.

A.5 Examples

We present examples for all 4 label types {*False*; *Partial Truth*; *Opinion stated as fact*; *True*} present in our dataset: DNF-300. Please refer Table-4,5,6,7. An annotated example from CDC dataset is displayed in Figure-7

Headline: Clinton Received Debate Questions Week Before Debate
The first presidential debate was held and Hillary Clinton was proclaimed the winner by the media. Indeed Clinton was able to turn in a strong debate performance, but did she do so fairly? Multiple reports and leaked information from inside the Clinton camp claim that the Clinton campaign was given the entire set of debate questions an entire week before the actual debate. Earlier last week an NBC intern was seen hand delivering a package to Clintons campaign headquarters, according to sources. The package was not given to secretarial staff, as would normally happen, but the intern was instead ushered into the personal office of Clinton campaign manager Robert Mook. Members of the Clinton press corps from several media organizations were in attendance at the time, and a reporter from Fox News recognized the intern, but said he was initially confused because the NBC intern was dressed like a Fed Ex employee. The reporter from Fox questioned campaign staff about the intern, but campaign staff at first claimed ignorance and then claimed that it was just a Fed Ex employee who had already left. No reporters present who had seen the intern dressed as a Fed Ex employee go into Mooks office saw him leave by the same front entrance. The Fox reporter who recognized the intern also immediately looked outside of the campaign headquarters and noted that there were no Fed Ex vehicles parked outside. Clinton seemed to have scripted responses ready for every question she was asked at the first debate. She had facts and numbers memorized for specific questions that it is very doubtful she would have had without being furnished the questions beforehand. The entire mainstream media has specifically been trying to portray Trump as a racist and a poor candidate. By furnishing Clinton with the debate questions NBC certainly hoped to make Clinton appear much more knowledgeable and competent than Trump. And though it is unlikely that anyone will be able to conclusively prove that Clinton was given the debate questions, it seems both logical and likely.
Type: 0 (*False*) **Authors**:Baltimore Gazette **URLs**: http://www.freemarketcentral.com/index.php/post/2503/report-clinton-received-debate-questions-a-week-before-debate
Evidence: [https://www.snopes.com/fact-check/clinton-received-debate-questions-week-before-debate/, https://www.truthorfiction.com/hillary-clinton-received-debate-questions-advance/]
Reason: Based on Snopes rating 'False' which means 'the primary elements of a claim are demonstrably false.'

Table 4: An example on *False* type from DNF-300 dataset.

Headline: Allergens in Vaccines Are Causing Life-Threatening Food Allergies
It would probably surprise few people to hear that food allergies are increasingly common in U.S. children and around the world . According to one public health website , food allergies in children aged 0-17 in the U.S. increased by 50% from 1997 to 2011. Although food allergies are now so widespread as to have become almost normalized, it is important to realize that millions of American children and adults suffer from severe rapid-onset allergic reactions that can be life-threatening. Foods represent the most common cause of anaphylaxis among children and adolescents. The United Kingdom has witnessed a 700% increase in hospital admissions for anaphylaxis and a 500% increase in admissions for food allergy since 1990. The question that few are asking is why life-threatening food allergies have become so alarmingly pervasive. A 2015 open access case report by Vinu Arumugham in the Journal of Developing Drugs , entitled " Evidence that Food Proteins in Vaccines Cause the Development of Food Allergies and Its Implications for Vaccine Policy ," persuasively argues that allergens in vaccinesand specifically food proteinsmay be the elephant in the room. As Arumugham points out, scientists have known for over 100 years that injecting proteins into humans or animals causes immune system sensitization to those proteins. And, since the 1940s, researchers have confirmed that food proteins in vaccines can induce allergy in vaccine recipients. Arumugham is not the first to bring the vaccine-allergy link to the publics attention. Heather Fraser makes a powerful case for the role of vaccines in precipitating peanut allergies in her 2011 book, The Peanut Allergy Epidemic: Whats Causing It and How to Stop It.
Type: 1 (*Partial Truth*) **Authors**:Admin - Orissa **URLs**: galacticconnection.com
Evidence: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3890451/
Reason: The key claim is written in such a way so that it misleads people in thinking all the food related allergies in US are caused by vaccines. Found evidence which says these type of allergies are rare.

Table 5: An example on *Partial Truth* type from DNF-300 dataset.

Headline: George Soros: Trump Will Win Popular Vote by a Landslide but Clinton Victory a 'Done Deal'
In recent weeks, Democrats have attempted to paint Republican presidential nominee Donald J. Trump as a lunatic for claiming that the election is going to be rigged in favor of his Democratic rival, Hillary Clinton. Even Republican politicians and former politicians are telling Trump to knock off such talk. But, as usual, Trump's shrewdness and defiance of standard political decorum in which the "opposition" party merely rolls over and surrenders in the face of Democratic pressure is winning the day. None other than billionaire investor and longtime Democratic supporter George Soros has said that the fix is literally in for the election, in favor of Clinton no matter how much of the popular vote, and from which battleground states, Trump captures. As reported by Top Right News and other outlets, during a recent interview with Bloomberg News, Soros a Democrat mega-donor openly admitted that Trump will win the popular vote in a "landslide." However, he said that none of that would matter, because a President Hillary Clinton is already a "done deal." In the interview, which is now going viral, Soros says with certainty that Trump will take the popular vote, despite what the polls say now (which are completely rigged to oversample Democrats), but not the Electoral College, which will go to Clinton. When the reporter asks if that is already a "done deal" that Clinton will be our next president no matter what Soros says "yes," and nods his head. Is Soros just making a prediction out of overconfidence? Or does he truly know something most of us don't know?
Type: 2 (*Opinion Stated As Fact*) **Authors**:J. D. Heyes **URLs**: https://www.naturalnews.com/055789_George_Soros_Hillary_Clinton_electoral_college.html
Evidence: 1. https://www.snopes.com/fact-check/george-soros-trump-will-win-popular-vote-by-a-landslide-but-clinton-victory-a-done-deal/, 2. https://www.bloomberg.com/news/videos/2016-01-22/soros-clinton-to-win-popular-vote-in-landslide
Reason: The key claim is false (based on Snopes rating), however, the article also contains opinions stated as fact.

Table 6: An example on *Opinion stated as fact* type from DNF-300 dataset.

Headline: Donald Trump: Minnesota Has Suffered Enough Accepting Refugees
In a pitch to suspend the nations Syrian refugee program , Donald Trump said Minnesotans have "suffered enough" from accepting Somali immigrants into their state. "Here in Minnesota you have seen first hand the problems caused with faulty refugee vetting, with large numbers of Somali refugees coming into your state, without your knowledge, without your support or approval," Trump said at a Minneapolis rally Sunday afternoon. He said his administration would suspend the Syrian refugee program and not resettle refugees anywhere in the United States without support from the communities, while Hillary Clintons "plan will import generations of terrorism, extremism and radicalism into your schools and throughout your communities."
Type: 3 (*True*) **Authors**:Henry Wolff **URLs**: amren.com
Evidence: 1. https://time.com/4560078/donald-trump-minnesota-somali-refugees/, 2. https://www.buzzfeednews.com/article/claudiakoerner/trump-vs-somali-refugees
Reason: Evidence found to support key claims.

Table 7: An example on *True* type from DNF-300 dataset.

SUKHAN: Corpus of Hindi Shayaris annotated with Sentiment Polarity Information

Salil Aggarwal **Abhigyan Ghosh** **Radhika Mamidi**

Language Technologies Research Centre
KCIS, IIIT Hyderabad
Telangana, India
salil.aggarwal@research.iiit.ac.in
abhigyan.ghosh@research.iiit.ac.in
radhika.mamidi@iiit.ac.in

Abstract

Shayari is a form of poetry mainly popular in the Indian subcontinent, in which the poet expresses his emotions and feelings in a very poetic manner. It is one of the best ways to express our thoughts and opinions. Therefore, it is of prime importance to have an annotated corpus of Hindi shayaris for the task of sentiment analysis. In this paper, we introduce SUKHAN, a dataset consisting of Hindi shayaris along with sentiment polarity labels. To the best of our knowledge, this is the first corpus of Hindi shayaris annotated with sentiment polarity information. This corpus contains a total of 733 Hindi shayaris of various genres. Also, this dataset is of utmost value as all the annotation is done manually by five annotators and this makes it a very rich dataset for training purposes. This annotated corpus is also used to build baseline sentiment classification models using machine learning techniques.

1 Introduction

Sentiment analysis is simply '*the task of extraction and analysis of subjective information present in some natural language data with the use of natural language processing*' [1]. But, the task of sentiment analysis becomes challenging for languages having annotated corpus only in some limited domains. One such language is Hindi and shayari is one of its domains which has no annotated dataset for the task of sentiment analysis. Shayari is a very rich tradition in South Asia. It has generally 2 to 4 lines which have some kind of deep meaning in them. It is mainly written in languages like Hindi, Urdu, Bangla, Nepali and Punjabi. Whether you are sad, alone, happy or in love, you can use shayari to express your feelings and thoughts. That's why, it is very important to have an annotated corpus of shayaris for the task of sentiment analysis. No such annotated corpus of Hindi shayaris currently exists in literature. SUKHAN is the first corpus of Hindi shayaris with annotated sentiment polarity information existing in literature as per our knowledge. It is written in Devanagari script and hence avoids the pre-processing cost of text normalization. We have also conducted various baseline experiments in order to compare the performance of various classifiers on the annotated corpus.

This paper is divided into 5 sections. Section 2 discusses related work in this area. Section 3 elaborates on the source and the creation of the corpus. Section 4 elaborates on the annotation process and the annotation scheme used for getting sentiment labels. Inter-annotator agreement has also been calculated and the details for interpretation of values for the Fleiss Kappa index have also been mentioned. Section 5 describes the experimental setup for training a model using various classifiers which helps in establishing the baseline for sentiment classification of these Hindi shayaris. All the results and conclusions using the annotated corpora have been briefly discussed in Section 6.

2 Related Work

So far, no work has been done to run sentiment analysis on Shayaris, neither in Hindi nor in any other Indian language, where similar constructs occur. However, work on sentiment analysis of Hindi poems has been done previously (Pal and Patel, 2020) but never on shayaris. Sentiment Analysis has also been done on Odia poems by Gaurav Mohanty and Mamidi (2018). Music classification has been carried out using lyrics (Hu et al., 2009), audio (Lu et al., 2005) and even multi-modal features (Laurier et al., 2009) for

[1] Source: Wikipedia

Proceedings of the 17th International Conference on Natural Language Processing, pages 228–233
Patna, India, December 18 - 21, 2020.

English. Similar work has been carried out for mood classification of Telugu (Abburi et al., 2016) and Hindi songs (Patra et al., 2016)

Nowadays, research mainly focuses on social media and very little attention is given to the traditional literature of which shayari is an important part. Automatic analysis of poetry is done for poems written in various languages like English, Chinese, Arabic, Malay, and Spanish. Barros et al. (2013) tried to categorize poems based on their emotional content. In the case of traditional literary works such as poetry, a lexicon creation methodology has been discussed for analyzing classical Chinese poetry (Hou and Frank, 2015). Hamidi et al. (2009) has also proposed a meter classification system for Persian poems based on features extracted from uttered poem. Alsharif et al. (2013) tried to classify Arabic poetry according to emotion associated with it.

3 Dataset

Due to the unavailability of annotated Hindi Shayari corpus with sentiment polarity information, the dataset was constructed manually. The advent of UTF-8 encoding has led to text in Indian scripts increasing day by day on the web. We collected shayaris from numerous online sources such as:

https://poetrytadka.com,

https://shayarifm.com/

https://shayarilovers.in/

https://bestnow.in/

These websites consist of many Hindi shayaris from various categories. We have used Hindi shayaris written using Devanagari script only. A total of 845 shayaris were mined. We have not used the title of shayari because the title of shayari usually does not have a strong association with the theme of the shayari. That's why, title was not used for extracting emotions from shayaris in order to avoid wrong results. The name of the shayar[2] also do not carry any sentiment information and therefore does not serve the task at hand and therefore is not used in baseline experiments. **Table 1** provides details on the initial statistics of the dataset before annotation.

[2]Person who writes shayaris

4 Annotation

4.1 Principles of Annotation

Three levels of granularity are described for existing methods of sentiment analysis. So, the task of sentiment analysis can be carried out at three different levels (Liu, 2012). On the basis of the level defined, the task is to identify if positive or negative sentiment is expressed at that level. It can be done at an aspect level (Hu and Liu, 2004), sentence level or at the level of the whole document (Turney, 2002). In the case of shayaris, it is possible that the different parts of the shayaris elicit different emotions. Since the task is to identify sentiment of the shayari as a whole, annotation is carried out only at an overall document level. The annotators were asked to go through the whole shayari before tagging them. This results in the tag corresponding to the polarity of the general mood evoked by it.

4.2 Annotation Process

We hired 5 annotators from different parts of India for the process of annotating the shayaris. These annotators were chosen from different regions in order to eliminate the chances of any kind of regional bias. These annotators were university students in the age group of 20–24 and were native Hindi speakers with sound background in linguistics and they speak and write in Hindi language on a daily basis. Each Hindi shayari was annotated by these 5 annotators. Each annotator was provided with the corpus and they had to annotate each and every shayari independently. They were not allowed to have any kind of communication with other annotators during the whole annotation process. Also, they were not given any kind of information regarding the shayar because there are some shayars who only write some particular type of shayaris which mostly evoke some particular kind of emotion like love, anger, hatred etc in the reader's mind. So revealing the name of shayar might result in some preconditioned bias which could have resulted in wrong annotation.

Shayaris are a very sophisticated form of language. At the same time, they can generate different kinds of thoughts and emotions in the mind of reader. So, a proper method is required for annotating the shayaris based on these emotions. Here, Russel's Circumplex Model (Russell and Pratt, 1980) serves as an appropriate reference

Sr. No.	Description	Initial Value
1	Initial shayari Count	845
2	Total number of words (tokens)	18978
3	Average number of words (tokens) per shayari	~ 23
4	Total number of unique words	2851

Table 1: Initial Dataset Statistics

for emotion identification. In this model, human emotions are plotted on a 2D plane of sentiment polarity and arousal as shown in Figure 1. For a given poem, the identified emotions were collected and based on these emotions, the sentiment polarity of the shayaris was decided. Initially, shayaris were classified into 5 categories:

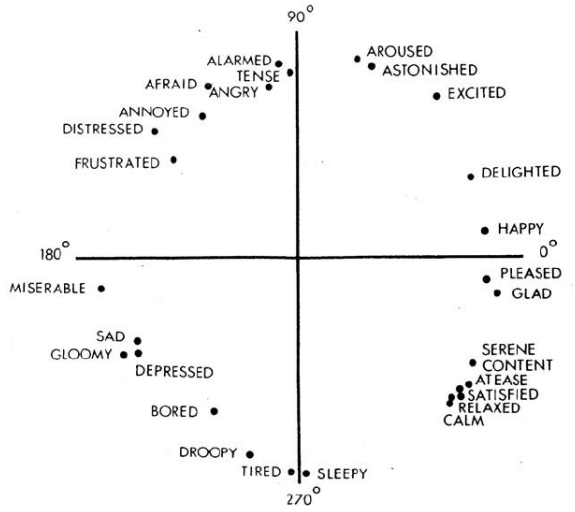


Figure 1: Russell's Circumplex Model of Affect

1. **Strongly Positive**: For the whole shayari, if the shayar is using only positive language such as expression of support, motivation, admiration, positive attitude, cheerfulness, forgiving nature, positive emotional state, etc, then the emotional states identified are tending to the positive side of Russell's model. These type of shayaris were classified as strongly positive.

2. **Strongly Negative**: For the whole shayari, if the shayar is using only negative language such as expression of hate, judgement, fear, anger, failure, criticism, negative attitude, etc. These types of shayaris were classified as strongly negative.

3. **Positive**: For the most of the shayari, if the shayar is using positive languages but also using negative language at some instants, then those types shayaris were classified as positive.

4. **Negative**: For the most of the shayari, if the shayar is using negative languages but also using positive language at some instances, then those shayaris were classified as negative.

5. **Neutral**: If the shayar is using both positive and negative language at equal intervals, then it is hard to tell what type of sentiment is present in the shayari. Those types of shayaris were classified as neutral.

Since the scope of this paper is to only determine a shayari as positive or negative, shayaris present in category 5 were discarded. Those present in Category 1 and 3 were annotated as positive and those present in Category 2 and 4 were annotated as negative. A total of 381 shayaris were annotated as positive whereas 352 shayaris were annotated as negative.

4.3 Inter-annotator Agreement

Inter-annotator agreement is a measure of how well the annotators can make the same annotation decision for the same category. Given the task in hand, it is fair to assume that annotation of the shayaris based on the emotions evoked by reading the lyrics is a very subjective opinion. Thus, inter-annotator agreement becomes an important factor in validating the annotators' work. The Fleiss' kappa obtained for the annotations for our dataset is 0.83. This corresponds to 'almost perfect agreement' according to the interpretation of Fleiss' kappa shown in **Table 2**.

5 Baseline for Sentiment Classification

In order to establish baseline results for the annotated corpus, a few experiments were conducted. The task was to classify Hindi shayaris as carrying positive or negative sentiment by training

Sr. No.	Range	Interpretation
1	$\leq$0	Poor agreement
2	0.01-0.20	Slight agreement
3	0.21-0.40	Fair agreement
4	0.41-0.60	Moderate agreement
5	0.61-0.80	Substantial agreement
6	0.81-1	Almost perfect agreement

Table 2: Fleiss Kappa values for inter-annotator agreement.

appropriate classification models. Initially three different classifiers were employed for this task and the results of each were compared. Term frequency-inverse document frequency (TF-IDF) (Jones, 1972) features were used to create a vector representation for an entire shayari. We also explored usage of character level n-grams as TF-IDF features to evaluate the performance of these classifiers.

5.1 Experimental Setup

The dataset was split into a ratio of 4:1 for the purpose of training and testing. For the baseline experiments, TF-IDF features for word n-grams and character n-grams were used for the task of sentiment classification. All the experiments were conducted using 'scikit-learn' (Pedregosa et al., 2011), an open source Python library[3]. Precision, Recall and F1-score are the three evaluation metrics which were calculated using **5-fold cross-validation**.

For baseline experiments Naive Bayes, Logistic Regression and Support Vector Machine (Cortes and Vapnik, 1995) were the classifiers used for baseline experiments. The 733 shayaris of the SUKHAN corpus were also passed through various pre-processing phases. Then using TF-IDF weights, vector representations were obtained for each shayari. TF-IDF is basically a technique to quantify a word in documents. We generally compute a weight to each word which signifies the importance of the word in the corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

[3]http://www.scikit-learn.org

6 Observations

TF-IDF was calculated for unigrams, uni-bigrams and uni-bi-trigrams. **Table 3** illustrates the results of the same for the three aforementioned classifiers. Even though 733 shayaris is a sizable corpus for the task in hand, it does not show a significant increase in accuracy especially with added bi-gram and tri-gram features. This is because most bi-grams and tri-grams occur sparsely in the entire corpus. Here, on an average, Linear-SVM performed better than all of the other classifiers. In order to tackle the problem of sparsity, we conducted experiments using n-grams at character level. For the baseline, 2-6 and 3-6 character n-grams were used to calculate character level TF-IDF features. The results of the same are illustrated in **Table 4**. On the basis of F1-score, Linear-SVM performed better than the other classifiers.

7 Conclusion

Shayaris and emotions share a very strong bonding. Each shayari is penned by a shayar with a lot of emotions, feelings and values. Different poetry elements such as diction, rhyme, rhythm, and imagery make shayari different from a normal piece of text. That's why In this research area, we have created a Hindi shayari corpus which would help to create an automatic system for categorization of shayaris based on polarity identification questionnaire and emotional states present in it. SUKHAN is the first corpus of Hindi shayaris of diverse themes, with shayaris, manually annotated as either having positive or negative sentiment values. The corpus was manually annotated with 733 Hindi shayaris scripted in the Devanagari script and based on the emotions they evoke. We also trained four different types of classifiers on our data. Classification models have been built using TF-IDF word-level features. Linear-SVM performed better as compared to other classifiers and the results of the experiments should serve as a good baseline.

Identifying the sentiment with the shayari is the first step towards identifying the emotions and thoughts which the shayari could evoke in the mind of the reader and this can further be used to build recommendation systems which are used by every major company in the e-commerce area.

Model	Features	Class	Precision	Recall	F1-Score
Linear-SVM	uni	Positive	0.852	**0.904**	0.876
		Negative	**0.907**	0.857	**0.880**
	uni-bi	Positive	**0.882**	0.859	0.870
		Negative	0.872	**0.896**	**0.884**
	uni-bi-tri	Positive	**0.888**	0.837	0.861
		Negative	0.856	**0.904**	**0.878**
Naive-Bayes	uni	Positive	**0.881**	0.850	0.864
		Negative	0.866	**0.896**	**0.880**
	uni-bi	Positive	0.869	0.871	0.870
		Negative	**0.882**	**0.880**	**0.881**
	uni-bi-tri	Positive	0.874	0.872	0.872
		Negative	**0.883**	**0.885**	**0.884**
Logistic regression	uni	Positive	**0.891**	0.819	0.853
		Negative	0.845	**0.909**	**0.875**
	uni-bi	Positive	**0.895**	0.810	0.850
		Negative	0.839	**0.914**	**0.875**
	uni-bi-tri	Positive	**0.898**	0.803	0.847
		Negative	0.834	**0.917**	**0.872**

Table 3: Sentiment Analysis with Word-Level TF-IDF Features

Model	Features	Class	Precision	Recall	F1-Score
Linear-SVM	(2-6) gram	Positive	0.873	0.864	0.868
		Negative	**0.876**	**0.886**	**0.880**
	(3-6) gram	Positive	**0.876**	0.862	0.868
		Negative	0.873	**0.888**	**0.880**
Naive-Bayes	(2-6) gram	Positive	**0.892**	0.808	0.845
		Negative	0.836	**0.901**	**0.869**
	(3-6) gram	Positive	**0.877**	0.819	0.844
		Negative	0.841	**0.893**	**0.865**
Logistic Regression	(2-6) gram	Positive	**0.882**	0.831	0.855
		Negative	0.851	**0.898**	**0.873**
	(3-6) gram	Positive	**0.880**	0.832	0.853
		Negative	0.852	**0.896**	**0.872**

Table 4: Sentiment analysis with Character-Level TF-IDF Features

References

Harika Abburi, Eswar Sai Akhil Akkireddy, Suryakanth Gangashetti, and Radhika Mamidi. 2016. Multimodal sentiment analysis of telugu songs. In *SAAIP@ IJCAI*, pages 48–52.

Ouais Alsharif, Deema Alshamaa, and Nada Ghneim. 2013. Emotion classification in arabic poetry using machine learning. *International Journal of Computer Applications*, 65(16).

Linda Barros, Pilar Rodriguez, and Alvaro Ortigosa. 2013. Automatic classification of literature pieces by emotion detection: A study on quevedo's poetry. In *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, pages 141–146. IEEE.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Pruthwik Mishra Gaurav Mohanty and Radhika Mamidi. 2018. Kabithaa: An annotated corpus of odia poems with sentiment polarity information. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France. European Language Resources Association (ELRA).

Saeid Hamidi, Farbod Razzazi, and Masoumeh P Ghaemmaghami. 2009. Automatic meter classification in persian poetries using support vector machines. In *2009 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pages 563–567. IEEE.

Yufang Hou and Anette Frank. 2015. Analyzing sentiment in classical chinese poetry. In *Proceedings of the 9th SIGHUM Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH)*, pages 15–24.

Minqing Hu and Bing Liu. 2004. Mining opinion features in customer reviews. In *AAAI*, volume 4, pages 755–760.

Yajie Hu, Xiaoou Chen, and Deshun Yang. 2009. Lyric-based song emotion detection with affective lexicon and fuzzy clustering method. In *ISMIR*, pages 123–128.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*.

Cyril Laurier, Mohamed Sordo, Joan Serra, and Perfecto Herrera. 2009. Music mood representations from social tags. In *ISMIR*, pages 381–386.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Lie Lu, Dan Liu, and Hong-Jiang Zhang. 2005. Automatic mood detection and tracking of music audio signals. *IEEE Transactions on audio, speech, and language processing*, 14(1):5–18.

Kaushika Pal and Biraj V Patel. 2020. Model for classification of poems in hindi language based on ras. In *Smart Systems and IoT: Innovations in Computing*, pages 655–661. Springer.

Braja Gopal Patra, Dipankar Das, and Sivaji Bandyopadhyay. 2016. Multimodal mood classification framework for hindi songs. *Computación y Sistemas*, 20(3):515–526.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

James A Russell and Geraldine Pratt. 1980. A description of the affective quality attributed to environments. *Journal of personality and social psychology*, 38(2):311.

Peter D Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. *arXiv preprint cs/0212032*.

Improving Neural Machine Translation for Sanskrit-English

Ravneet Punia, Aditya Sharma, Sarthak Pruthi, Minni Jain
Delhi Technological University
Delhi, India
{ravneetpunia_bt2k16,adityasharma_bt2k17,sarthakpruthi_bt2k18it110,minnijain}@dtu.ac.in

Abstract

Sanskrit is one of the oldest languages of the Asian Subcontinent that fell out of common usage around 600 B.C. In this paper, we attempt to translate Sanskrit to English using Neural Machine Translation approaches based on Reinforcement Learning and Transfer learning that were never tried and tested on Sanskrit. Along with the paper, we also release monolingual Sanskrit and parallel aligned Sanskrit-English corpora for the research community. Our methodologies outperform the previous approaches applied to Sanskrit by various researchers and will further help the linguistic community to accelerate the costly and time-consuming manual translation process.

1 Introduction

Sanskrit is one of the oldest, extensively studied, and researched languages in the world.[1] It is the oldest Indo-Aryan Language prominently used in Indo-European studies and now used for interlingual translation to English and many other Indian languages, however, the fact that it is dead in today's time cannot be denied. English has emerged as the most popular language on the world level, and the advent of globalization has led to the need for cross-language translations. The developing regions still used the regional languages, and thus the translation of the English language into local languages can make information accessible. Machine Translation is one of the most onerous tasks in natural language processing. Sanskrit is unique as it does not work using a noun-phrase model.[2] It's strict grammar rules, and syllables match being a direct parent of Modern Hindi language. The challenges faced during machine translation of Sanskrit to other languages are translation divergence or the ambiguity phrase due to multiple-meaning, the lack of parallel language data. Lots of historical and cultural data such as Bhagavad Gita, Ramayana, Mahabharata, and Hindu Literature Vedas were originally written in the Sanskrit language, and most of them are untranslated to other languages. Despite its important part in Indian culture and history, not much work has been done for translation to or from the Sanskrit language.

Although the past few years, many efforts have been made to translate Sanskrit to other languages using various machine translation approaches. Mishra and Mishra (2008) and Gupta et al. (2013) implemented example-based and rule-based approaches for Sanskrit-English machine translation. Later Mishra and Mishra (2010) improved the Rule-Based Machine Translation approach by integrating with the Artificial Neural Network (ANN) model. Recently Koul and Manvi (2019) proposed an encoder-decoder based Neural Machine Transition approach for Sanskrit to English.

In recent years Neural Machine Translation techniques like Sequence to Sequence Learning, Encoder-Decoder attention-based architectures (Bahdanau et al., 2014), and Transformers have achieved State Of The Art (SOTA) results for supervised machine translation tasks. However, for low resource methods like Back translation (Edunov et al., 2018), Cross-Language Modeling, Phrase-Based Machine Translation (Lample et al., 2018), and Dual Learning Mechanism based upon reinforcement learning(He et al., 2016) takes the benefit of monolingual data to improve the quality of translations over supervised approaches. Unfortunately, none of the above methods has been used for Sanskrit's machine translation task due to the lack of linguistic resources.

Through this paper

[1] https://en.wikipedia.org/wiki/Sanskrit

[2] https://www.genpact.com/

Proceedings of the 17th International Conference on Natural Language Processing, pages 234–238
Patna, India, December 18 - 21, 2020.

- We test multiple machine translation approach based supervised methodology, Transfer Learning, and reinforcement learning approach that leverages monolingual data for Neural Machine Translation (NMT).

- We also release the collected parallel English - Sanskrit data as well as monolingual data for Sanskrit.

2 Related Work

Work by Mishra and Mishra (2009) mainly focuses on building tokenization, POS Tagger, and a Named Entity Recognition (NER) system for the Sanskrit language using statistical machine translation approach. Mane et al. (2010) introduced a dictionary-based approach for implementing machine translation on Sanskrit by parsing and replacing source word with the target using a bilingual dictionary.

Bahadur et al. (2012) developed Machine translation which primarily focused formulation of Synchronous Context-Free Grammar (SCFG) and a subset of Context-Free Grammar (CFG). The developed model firstly tokenize input data and then match the exact word or phrase from the dictionary. The developed model also gathers information about parts of speech (POS) of input sentences. The work by Rathod (2014) implemented a Rule-Based and Example-based approach for Machine translation using a bilingual dictionary and speech synthesizer that also converts speech to text. The designed model was capable of grammar and spell check too. An open-source web portal [3] collects data from domains like primary and secondary school Sanskrit literature books, also established by Govt. of India in 2015. It also implements statistical Machine Translation algorithms and even tries to solve Word Sense Disambiguation (WSD) problem.

Apart from Koul and Manvi (2019) encoder-decoder model, no such work has been done on Sanskrit's Neural Machine Translation in the best of author's knowledge.

3 Dataset

For this paper, we extracted parallelly allied English-Sanskrit data as well as monolingual data for each language. The parallel English-Sanskrit data, we obtained 2,100 sentences from OPUS[4], Sanskrit translation of Bible, Shlokas from Ramayana and more sentences from Gita. As data is extracted from multiple sources, sentences with the same source but multiple translations and sentences with the same translation, various sources are removed. Finally, a parallel dataset with 9000 parallel lines is extracted, further divided into the standard train, test, and validation set with a ratio of 80:10:10, respectively.

For the monolingual data, we collected the data from the Romanized version of Mahabharata, consisting of 130,000 lines (approx) and for English, we extracted Europarl dataset (Koehn, 2005)

4 Proposed Methodology

Previous Neural Machine Translation approaches for Sanskrit mainly focus on Rule-Based Approach and Encoder-Decoder Mechanism using LSTM units. The classical Rule-Based approach is time-consuming, requires much manual work by the linguist, and does not have good learning capabilities. In contrast, LSTMs based models tend to overfit faster, suffer from issues related to polysemy, and multiple word senses (Calvo et al., 2019; Huang et al., 2011).

To handle all these issues, we first established a baseline translation model using a multi-head self-attention mechanism using encoder-decoder architecture, as suggested by Vaswani et al. (2017). Further, we improved the baseline translator using a reinforcement learning approach by establishing language models and agents that leverage monolingual data. We further experimented with the Transfer Learning approach for Machine Translation to get the benefit of lexically similar Hindi language that is rich resource language.

4.1 Transformer Translator

Initially, the raw sentences were tokenized using SentancePiece tokenizer (Kudo and Richardson, 2018), which is an unsupervised and language-independent tokenization method. Further, the parallel and monolingual tokenized data was used to train word-vectors of length 128, using the word2vec (Mikolov et al., 2013) technique. As the transformer architecture doesn't maintain any word order, so along with the trained word-vectors, a positional encoding signal is mixed and given to the encoder as input. Introducing the positional

[3] http://sanskrit.jnu.ac.in/shmt/index.jsp

[4] http://opus.nlpl.eu/

encoding helps maintain the embedding information and gives the vital position information to the encoder. In the architecture, both encoder and decoder are formed by stacking four identical layers in the same manner as described by Vaswani et al. (2017). The encoder takes the representation of Sanskrit token through word embedding and positional encoding, which is then fed to a multi-head attention unit where feed-forward units with residual connections are employed between every other sublayer. This signal normalizes and is given to the decoder as input along with the output embeddings, positional encoding, and masked multi-head attention. The decoder works similar to the encoder and generates output word by word and finally makes a sequence.

4.2 Reinforcement Translator

This methodology is inspired by He et al. (2016), where we used our Transformer model as the translation model, and building the language model from the Recurrent Neural Network (RNN) using the monolingual data only. We define dual NMT as a combination of Sanskrit to English considered to be the primary task and English to Sanskrit being dual. For both primary and dual tasks, we set individual agents to perform two agent communication games where they correct each other through a reinforcement learning process. The reward system is a combination of Language model (r_1) reward and communication reward (r_2), which can be expressed using the equation:

$$r = \alpha(r_1) + (1 - \alpha)r_2. \quad (1)$$

Where α is a hyper-parameter which is set to 0.1. Further Transformer models are improved using a policy gradient method (Sutton et al., 2000) for maximum reward, which is a common methodology in reinforcement learning. The process iterated for 600 rounds and stoped when the translation model converges. Other parameters such as beam search size, learning rate, the individual reward for each agent r_1 and r_2 were taken same as defined by He et al. (2016)

4.3 Transfer Learning

The main idea of transfer learning is to transfer the knowledge learned by a model trained on a high resource language set, i.e., parent model, to train another model with a similar application, i.e., child model. For our experimentation, we firstly prepared a Hindi-English NMT model using Transformers, as the parent model on 1.56 Million parallel data provided by Kunchukuttan et al. (2017) and training Sanskrit - English NMT model as a child model. The Hindi data was firstly tokenized using Indic Tokenizer (Kunchukuttan, 2020), English using Moses tokenizer (Koehn et al., 2007), and Sanskrit using sentencepiece (Kudo and Richardson, 2018). Further Hindi-English NMT model was trained using the same training procedure as of Transformer model discussed in section 4.1

5 Result & Discussion

The baseline model in section 4.1 was implemented using the OpenNMT Framework (Klein et al., 2017). For the transfer learning implementation, we used the NEMATUS toolkit (Sennrich et al., 2017). The baseline and child model in the transfer learning approach were trained, tuned, and tested on the same data split set discussed in section 3. For the quantitative evaluation, we used the BLEU score (Papineni et al., 2002) for English translation generated by the model against the test set. The results obtained are shown in Table 1.

Architecture	BLEU	Rating
1. Transformer Translator	4.6	2.4
2. Reinforcement Translator	5.8	2.9
3. Transfer Learning	18.4	3.9

Table 1: Evaluation of different models with English translation using BLEU scores

For the qualitative analysis, five Sanskrit language experts were randomly given 50 sentences each from all three models for the rating based on the following rating schema:

- Good[5]: Sentence is interpretable by the language expert, having no incorrectly translated words.
- Helpful[3]: Sentence is interpretable by the language expert with some context knowledge, has some errors and wrong word order.
- Partially Helpful[1]: contains incorrectly translated content words, few UNK Tokens, but still interpretable by language experts.
- Wrong[0]: Sentence having many UNK Tokens or untranslated words and considered as not translated by a language expert.

All average ratings are shown in the last column of Table 1. Hyperparameters searched and best selected for the baseline model during the training are mentioned in the Table 2.

Hyperparameters	Experimented	Best
Epochs	200	200
Batch-Size	512,1024	1024
Number of Layer	4	4
Learning Rate	Dynamic	
Dropout	0.1	0.1
Dimensional Vectors	128,256	128

Table 2: Hyperparameter searching for the best results

Few Observations from results:

- Transfer Learning approach performs best among all three models. The lexical similarity between Hindi and Sanskrit helped in achieving a better result.
- Transformer translator performed worst, most likely due to small and sparse dataset from various domains and a large number of parameters of the model. However, Reinforcement learning made a slight improvement of 1.2 BLEU points.
- The dataset used Koul and Manvi (2019) is different and not available to the public domain for testing, so it won't be appropriate to compare results of Koul and Manvi (2019) with our experiments.

6 Conclusion

In this paper, we explored approaches that have never before been used for the translation of the Sanskrit language to English. Firstly we established a baseline with the Transformer architecture. Further, we improved the Transformer model with Dual Learning methodology and gained small improvement on BLEU Score. The best BLEU Score we observed was with the Transfer Learning method. Although we will not like to make an explicit comment that Transformers architecture is the first time explored in our research, a few unofficial repositories have worked and published the results. In the future, we would try to add more parallel data to improve the trained models' quality. We believe that our research would open the doors for many researchers, linguists, and students to work and explore Sanskrit.

Dataset, training subroutine, and trained model is available at: https://github.com/RavneetDTU/Improving-Neural-Machine-Translation-for-Sanskrit-English

References

Promila Bahadur, AK Jain, and DS Chauhan. 2012. Etrans-a complete framework for english to sanskrit machine translation. In *International Journal of Advanced Computer Science and Applications (IJACSA) from International Conference and workshop on Emerging Trends in Technology*. Citeseer.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Hiram Calvo, Arturo P Rocha-Ramirez, Marco A Moreno-Armendáriz, and Carlos A Duchanoy. 2019. Toward universal word sense disambiguation using deep neural networks. *IEEE Access*, 7:60264–60275.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*.

V. K. Gupta, N. Tapaswi, and S. Jain. 2013. Knowledge representation of grammatical constructs of sanskrit language using rule based sanskrit language to english language machine translation. In *2013 International Conference on Advances in Technology and Engineering (ICATE)*, pages 1–5.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in neural information processing systems*, pages 820–828.

Fei Huang, Alexander Yates, Arun Ahuja, and Doug Downey. 2011. Language models as representations for weakly supervised nlp tasks. In *Proceedings of the fifteenth conference on computational natural language learning*, pages 125–134.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86. Citeseer.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL*

on interactive poster and demonstration sessions, pages 177–180. Association for Computational Linguistics.

Nimrita Koul and Sunilkumar S Manvi. 2019. A proposed model for neural machine translation of sanskrit into english. *International Journal of Information Technology*, pages 1–7.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Anoop Kunchukuttan. 2020. The IndicNLP Library. https://github.com/anoopkunchukuttan/indic_nlp_library/blob/master/docs/indicnlp.pdf.

Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. 2017. The iit bombay english-hindi parallel corpus. *arXiv preprint arXiv:1710.02855*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

DT Mane, PR Devale, and SD SURYAWANS. 2010. A design towards english to sanskrit machine translatio and sy thesizer syste 1 si grule base approach. *Int J Multidisp Res Adv Eng*, 1(1).

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Vimal Mishra and RB Mishra. 2008. Study of example based english to sanskrit machine translation. *Polibits*, (37):43–54.

Vimal Mishra and RB Mishra. 2009. Divergence patterns between english and sanskrit machine translation. *INFOCOMP Journal of Computer Science*, 8(3):62–71.

Vimal Mishra and RB Mishra. 2010. Approach of english to sanskrit machine translation based on case-based reasoning, artificial neural networks and translation rules. *International Journal of Knowledge Engineering and Soft Data Paradigms*, 2(4):328–348.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Sarita G Rathod. 2014. Machine translation of natural language using different approaches. *International journal of computer applications*, 102(15).

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nădejde. 2017. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, Valencia, Spain. Association for Computational Linguistics.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Parsing Indian English News Headlines

Samapika Roy[1], **Sukhada**[1], **Anil Kr. Singh**[2]
[1]Dept. of Humanistic Studies, IIT (BHU)
[2]Dept. of Computer Science and Engg., IIT (BHU)
{samapikar.rs.hss15, sukhada.hss, aksingh.cse}@itbhu.ac.in

Abstract

Parsing news headlines is one of the difficult tasks of Natural Language Processing (NLP). It is mostly because news headlines (NHs) are not complete grammatical sentences. News editors use all sorts of tricks to grab readers' attention. For instance, unusual capitalization as in headline 'Ear SHOT ashok rajagopalan'; some demand world knowledge like 'Church reformation celebrated' where 'Church reformation' refers to a historical event and not a piece of news about an ordinary church. The lack of transparency in NHs can be linguistic, cultural, social, or contextual. The lack of space provided for a news headline has led to creative liberty.

Though many works like news value extraction, summary generation, emotion classification of NHs have been going on, parsing them had been a tough challenge. Linguists have also been interested in NHs for creativity in the language used by bending traditional grammar rules. Researchers have conducted studies on news reportage, discourse analysis of NHs, and many more. While the creativity seen in NHs is fascinating for language researchers, it poses a computational challenge for NLP researchers. This paper presents an outline of the ongoing doctoral research on the parsing of Indian English NHs. The ultimate aim of this research is to provide a module that will generate correctly parsed NHs. The intention is to enhance the broad applicability of newspaper corpus for future NLP applications.

1 Introduction

NHs stands to be an excellent example of creative writing. Headlines tend to be short, attention-grabbing, and giving out just enough information to attract readers' attention. To keep readers engage, news editors use all sorts of tricks possible. Studies on Indian English have been going on for some decades and being a part of South Asian English, Indian English has managed to grab attention for the past few years. Like other stratification of the English language, Indian English is unique as well. Though it follows the basic underlying structure of British English, the syntax, semantics, and pronunciation of Indian English differ from world English. Translating from native language to English sometimes results in different sentence structures. Due to linguistic diversity, code-mixing and code-switching are extremely common. These effects can be found in NHs as well.

Sometimes news editors tend to use local idiosyncrasies. NHs are different from standard text. For example, we can find unusual capitalization as in 'Ear SHOT ashok rajagopalan', deliberate subject noun phrase drop as in 'Arrested for theft', the deliberate dropping of main verb as in 'Identity cards for all urban street vendors', dropping of auxiliaries as in '18th century stone inscription unearthed', so on. Such structures are unique to NHs which make it difficult to parse them using existing parsers. This research attempts to solve such problem of NHs parsing.

2 Contributions

This research contributes at following levels :

1. NHs Corpus: A corpus of 40,000 (approx.) headlines containing 3 lakh words (approx.) of Indian English NHs has been collected.
2. Parallel corpus: A parallel corpus of NHs and grammatically transformed corresponding sentences has been created.
3. Linguistic analysis of the NHs data: Through the Linguistic analysis, different structures of NHs, words compositions, voices, tenses, dropping of subjects, as well as usage of punctuation have been observed.

Proceedings of the 17th International Conference on Natural Language Processing, pages 239–242
Patna, India, December 18 - 21, 2020.

4. Guideline creation: A proper set of guidelines for the transformation of NHs has been drafted covering the various structures of NHs that were found out as a result of linguistic analysis,
5. Feature model: We have created a syntactico-semantic feature model based on linguistic analysis conducted on NHs corpus.
6. Headline grammar: The creation of news headline grammar based on the linguistic analysis is going on.
7. NHs Module: Creation of a module which could provide us correct parsing of NHs is in the process.

3 Methodology

For the study, we deliberately chose Indian English and collected data for two reasons: 1) apart from the linguistic analysis, a contrastive analysis between NHs of Indian English and British English has been conducted. This study helped us to understand whether the structure of NHs in Indian English is different from NHs in British English and other English varieties, 2) Working on Indian English, collecting Indian English NHs data and building a parallel corpus will be fruitful for future endeavors as some computational works have been already conducted on Indian English. On the other hand, Indian English data collection is going on for quite a some time in many organizations.

We collected data from three different newspapers to analyze the syntactic structures. The lack of space in hard copy newspapers leads to an opportunity for creative liberty for news editors, like excluding grammatical and lexical elements (dropping subject noun phrase, auxiliary drop), the-out-of-grammar use of punctuation, and so on. For a systematic analysis of the data, we adhered to an inductive research strategy.

Our very first objective was to analyze the data for a better understanding of the problem. This analysis helped us to come up with an idea about accuracy of the currently available parsers and areas where parsing output can be improved. After parsing (Constituency) the data with existing open-source parsers like Stanford and Allennlp, we observed several errors like parsing of singular verbs as plural nouns, adjectives as verbs, and so on. There was a need to understand the structures of NHs and thus to analyze the NHs corpus linguistically. This study was exploratory and interpretive. For this we used both qualitative and quantitative methods for the data analysis.

We found out that NHs can be either complete headlines (headlines following the necessary SVO word order of English language) and fragments (comprises mostly of noun phrases, ex. 'A burning issue'). The complete NHs were further divided into four categories according to structures: declarative, imperative, interrogative, and exclamatory. The grammatical notions we used to describe the headlines are in a broader sense as headlines can not fulfill every linguistic need to belong in any particular category. Instead, they fulfilled mere basic needs. The linguistic analysis of NHs helped us create a grammar and guideline to transform NHs to their canonical form to get the correct parse.

Based on the linguistic analysis, we have build a syntax and semantics-based feature model. In this model, we have tried to cover the various headlines' structures we have come across so far in our study. It has been created with the motive that it will help us in the annotation of the NHs. Also, it will work as a guideline to create a transfer grammar analysis module. In the feature model, we covered all the aspects of NHs like headlines structure, which can be a complete NH or a fragment; there are NHs with or without subject noun phrase; NHs which is only a quote from a speaker but without the mention of the speaker, and so on. Our next step is to automate the transformation. We decided to treat the problem as a machine translation problem, for which we are working on creating a parallel corpus of raw NHs and equivalent grammatically transformed sentences. We are also working on creating transfer grammar for this purpose.

4 Observations

We have observed following specific vital issues during the research so far:

1. Incorrect parsing: We observed that parsing the NHs with the current available parsers is difficult as some grammatical elements are intentionally dropped in NHs. The error analysis helped us to identify the structural issues of parsing the NHs with existing parsers. In order to understand the syntax of NHs, the linguistic analysis of the NHs corpus became necessary. For example, Nouns marked as Verbs: (ROOT (S (NP (CD Two) (NN policemen)) (VP (VBD suspended) (SBAR (IN as) (S (VP (VBD

accused) **(NP (NNS escapes)**) (PP (IN from) (NP (NN custody)))))))))

2. Abridged structure: The linguistic analysis of the NHs gave us an insight into the internal structure of NHs which leads to a grammar which is specific to headlines.
3. Limited information: We faced specific issues while transforming the NHs into grammatical sentences. For transformations, our intention was not to add anything that is not certain from the NHs as well as be careful in not removing any vital information from the NHs. With the limited information we can get from the NHs, we focused on fulfilling the basic yet essential grammatical requirements.
4. Approach: Since there is no existing framework from which we can draw inspirations, we decided to go with hybrid approach in creating our module for parsing the NHs.

5 Results

The results of the error analysis of the data after constituency parsing, shows that incorrect tags e.g. plural nouns as an adjective, singular verbs as plural nouns, common nouns as proper nouns and so on led to incorrect parsed output. Following is an example of such output which shows multiple parsing errors: (ROOT(NP(NP **(NNP Boat)**)(NP **(JJ capsize)** (NN toll) **(NNS touches)**) (NP (CD 21)))). As we can observe here, 'Boat' has been incorrectly parsed as NNP(proper noun)'capsize' above is incorrectly parsed as JJ (adjective) and 'touches', above is incorrectly tagged as NNS (singular noun). The reason behind these incorrect tags is the intentional dropping of grammatical elements by editors due to space constraints. The errors occurred mostly because these NHs are not grammatically structured. The errors are the lack of crucial grammatical elements in the NHs, which distorts the grammatical bond in a sentence.

To cross-examine our observations, we provided the parsers with sample data of grammatically transformed raw NHs, where we added a few essential grammatical elements required. Proving our theory (which we developed from error analysis) correct, the parsers provided the correct parse. This finding has important implications for developing the module we intend for the generation of correctly parsed NHs. In linguistic analysis, we observed specific structural constructions like dropping off the subject and out of the grammar style of using linguistic devices like punctuation, idioms, multiword expressions, and many more.

6 Future Goals

Future research will be devoted to the development of news headline grammar and on the viability of our approach. We have already conducted linguistic analysis, which will act as a framework for headline grammar. We are currently working on automating the NHs corpus to the standard canonical sentences. We intend to elaborate the research on the contrastive analysis between a parallel corpus of raw headlines and its grammatically transformed sentences. For the rule-based approach, we are moving forward with the creation of context-free-grammar rules. For automating the NHs to its grammatical form, we are considering LALR parsers to start. In the meantime, we are trying to create enough parallel corpus of the raw NHs corpus we collected and the grammatically transformed sentences of those NHs to go with a statistical approach.

7 Research Roadmap

The research road map is as follows: We did data collection and sanitization during this research followed by constituency parsing of data, performed the error analysis, conducted the linguistic analysis of NHs, did manual transformations of NHs for the rule-based approach, and formulated a guideline for the transformations, and build a feature design model for the NHs. We are currently working on constructing a news headline grammar, automating the feature annotation process, and finally building a module to provide us with correctly parsed NHs as final output.

Acknowledgments

I would like to express my sincere gratitude to my supervisor Dr. Sukhada, Assistant Professor, dept. of Humanistic Studies, IIT (BHU) and my co-supervisor Dr. A.K. Singh, Associate professor, dept. of Computer Science and Engg., IIT (BHU) for their constant support and guidance.

References

Innocent Ejimofor Agu. 2015. A linguistic-stylistic analysis of newspaper reportage. *International Journal*, 20.

Alireza Bonyadi and Moses Samuel. 2013. Headlines in newspaper editorials: A contrastive study. *Sage Open*, 3(2):2158244013494863.

Christine Develotte and Elizabeth Rechniewski. 2001. Discourse analysis of newspaper headlines: a methodological framework for research into national representations. *The Web Journal of French Media Studies*, 4(1):1–12.

Daria Lombardi. 2018. Critical discourse analysis of online news headlines: A case of the stoneman douglas high school shooting.

Novriyanto Napu. 2018. English and indonesian newspaper headlines: A comparative study of lexical features. *European Journal of Literature, Language and Linguistics Studies*.

Alicja Piotrkowicz, VG Dimitrova, and Katja Markert. 2017. Automatic extraction of news values from headline text. In *Proceedings of the Student Research Workshop at the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL SRW 2017)*, pages 64–74. Association for Computational Linguistics.

Satoru Takahashi, Masakazu Takahashi, Hiroshi Takahashi, and Kazuhiko Tsuda. 2007. Analysis of the relation between stock price returns and headline news using text categorization. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 1339–1345. Springer.

Word Sense Disambiguation For Kashmiri Language Using Supervised Machine Learning

Tawseef Ahmad Mir[1] , Aadil Ahmad Lawaye[2]
Baba Ghulam Shah Badshah University Rajouri - Jammu & Kashmir – 185234
[1] tawseefmir1191@gmail.com
[2] aadil.lawaye@gmail.com

1 Introduction

Every language spoken by people in this world contain words that have more than one meaning. Meaning of such words at a particular time depends on the context in which the word has been used. The process of selecting the meaning of ambiguous word from the set of possible meanings is called word sense disambiguation (WSD). WSD is one of the hot research topics in the natural language processing (NLP) domain. For humans it seems to be very easy to understand the meaning of ambiguous words but it is a very complex problem for machines to do so. Consider the following sentences in English:

- This saw is blunt

  I saw a horror dream yesterday

In the first sentence the word saw is used as noun and its meaning is a tool used to cut hard material like wood, metal. In the second sentence the word saw is past form of verb see.
Similarly consider the following sentence in Kashmiri:

- Thave dare yel
  Open the window
- Rache daare zeeth.
  Keep long beard.

In the above two sentences the word *daare* is having two different meanings. In the first sentence it means window where as in the second sentence it means beard.

In NLP WSD is considered as an AI Complete problem, that is, a problem whose solution presumes a solution to understanding natural language or common-sense reasoning (Ide et al, 1998).The meaning of an ambiguous word depends heavily on the words surrounding it. To resolve the ambiguity of words number of approaches have been designed till date and work is still going on. The research on WSD actually started in 1940's making it one of the oldest problems in the computational linguistics. Some important research works for handling WSD in various languages are (Abid et al, 2017), (Borah et al, 2019), (Khaled et al, 2012), (Richard et al, 2014), (Basuki et al, 2019), (Rajat & Sudip , 2015), (Himdweep et al, 2017), (Tarjni & Amit, 2019). For resolving ambiguity in Kashmiri language no work is cited. The objective of this research is to propose the WSD for Kashmiri using Supervised Machine Learning approaches.

2 Motivation

The driving motivation for this research is that WSD is an intermediate step for the various NLP applications like Machine Translation, grammatical analysis, speech processing, Information Retrieval and hypertext navigation (Ide & Veronis, 1998) and developing efficient WSD system is very crucial for the better performance of these NLP applications.

Since there is no work cited in Kashmiri language for handling WSD problem this is the first attempt in this direction, so this also motivated me for this research.

The third point is that research in NLP applications for Kashmiri language is in infancy stage this research will boost the research in this field and attract researchers to work in this field of Artificial Intelligence.

3 Research Challenges

Word sense disambiguation is a computationally

Proceedings of the 17th International Conference on Natural Language Processing, pages 243–245
Patna, India, December 18 - 21, 2020.

complex task and poses a lot of difficulties to the researcher. As far as this study is concerned there are a number of challenges. Notable challenges include:

Resource Scarcity: Kashmiri language is a resource poor language as adequate resources are not available for research which makes our task difficult. Only work done in Kashmiri so far in this domain is the development of some corpus and few linguistic tools under the project "Development of Language Tools and Linguistic Resources for Kashmiri" at the Department of Linguistics, University of Kashmir (Aadil et al, 2009), (Aadil et al, 2009), (Aadil et al,2010),(Aadil et al, 2013) , (Aadil et al, 2012) , (Aadil et al, 2012), (Aadil et al,2011).

Sense selection: One important issue related to WSD is to select senses of ambiguous word as different sources provide different divisions of words into senses.

Inter-Judge Variance: This study is the first attempt towards resolving ambiguity in Kashmiri language so the only option to evaluate the WSD system for Kashmiri language is human-judgement. But different humans may give different meanings for the same word. This increases the complexity of WSD task.

Discreteness of senses: WordNet contain very fine-grained senses and often it is very difficult to differentiate between these senses. This causes the disagreements among the lexicographers to specify which senses should be considered different ones for a particular word.

3 Grammar Formalism and Issues specific to Kashmiri Language

Grammar formalism is of great importance for creating syntactic annotation corpus and frameworks available can be categorized into two types: Dependency based annotation scheme and Constituency based annotation. In constituent-based annotation scheme sentence is depicted as hierarchically organized phrases and relation between and within constituents is not represented explicitly. In dependency based annotation the sentence is organized as dependency graph consisting of a head and dependent with labelled arc specifying relationship between them.

Kashmiri language being inflectionally rich dependency annotation scheme existing for Hindi-Urdu is considered suitable for annotation. But some issues needed to be addressed. These issues include V2 phenomenon, discrepancy that exists between coordinating and subordinating conjuncts, rift in complex predicates, pronominal clitics etc.(Bhat , 2012).

5 Methodology

In this study Supervised Machine Learning approaches are to be used to handle WSD in Kashmiri language. The Supervised Machine learning approaches work in two phases i.e, training phase and test phase. In training phase classifier is trained how to resolve ambiguity of a polysemous word (words having multiple meanings). In the testing phase the classifier assigns most appropriate sense to ambiguous word. The flowchart below depicts the methodology to be used:

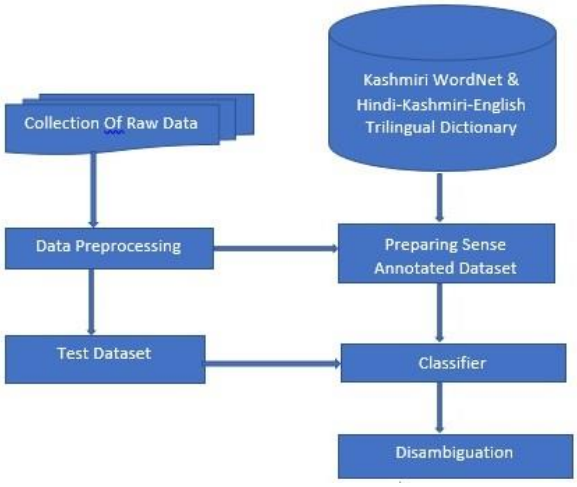


Fig 1. Proposed WSD System for Kashmiri

5.1 Collection of Raw Data

Data for this study will be collected from online (newspapers, blogs etc.) and offline resources. It's a very challenging task in this study.

5.2 Data Preprocessing

Usually the data is not readily available for research so it needs to be preprocessed to make it suitable for research. Data preprocessing usually involves stop word removal, data cleaning, stemming, removing inconsistencies in data.

5.3 Preparation of dataset

The data collected is divided into two sets. Training dataset and test dataset. The trained dataset sense tagged using Kashmiri WordNet and Hindi-Kashmiri-English Trilingual dictionary is used to

train the classifier so that it can disambiguate the ambiguous word for which it has been trained.

5.4 Classification

The supervised machine learning approach would be used to train the classifier and the classifier would be used to predict the meaning of the polysemous word in the test phase.

6 Experimental Outcomes

The main outcomes of the study are as follows:

1. Sense Annotated Corpus for Kashmiri Language.
2. WSD Data Set.
3. Word Sense Disambiguation System for Kashmir Language.

References

Aadil Amin Kak, Nazima Mehdi and Aadil Ahmad Lawaye 2009. What should be and What should not be? Developing a POS tagset for Kashmiri. Interdisciplinary Journal Of Linguistics (IJL), 2 185-196

Aadil Amin Kak, Nazima Mehdi and Aadil Ahmad Lawaye.. 2009. Towards Developing A Tagset For Kashmiri. Nepalese Linguistics, 49-60.

Aadil Amin Kak, Nazima Mehdi and Aadil Ahmad Lawaye 2010.Building a Cross Script Kashmiri Converter. Issues and solutions. *Proceedings of Oriental COCOSDA (The International Committee for the Co-ordination and Standardization of Speech Databases and Assessment Techniques)*.

Aadil Ahmad Lawaye and Prof. Bipul Syam Purkayastha2 2013. Towards Developing A Hierarchical Part Of Speech Tagger for Kashmiri: Hybrid Approach. *Proceedings of the 2nd National Conference on Advancements in the Era of Multidisciplinary Systems, Elsevier Publications, 187-192.*

Aadil Ahmad Lawaye, and Dixit N. 2012. Multilingual Dictionary Generation Using Indo-Wordnet:A Proposal', THE COMMUNICATIONS-Journal of Applied Research in Open and Distance Education, 188-191.

Aadil Ahmad Lawaye, Bipul Syam Purkayastha. 2014. *Kashmir Part of Speech Tagger Using CRF", Indian Journal of Research, 37-38*

Basuki, Setio, Ali Sofyan Kholimi, Agus Eko Minarno, Fauzi Dwi Setiawan Sumadi, and M. Rizal Arif Effendy. 2019 Word Sense Disambiguation (WSD) for Indonesian Homograph Word Meaning Determination by LESK Algorithm Application," *12th International Conference on Information & Communication Technology and System (ICTS), Surabaya, Indonesia, 2019,* pp. 8-15

Bhat, S.M., 2012, December. Introducing Kashmiri dependency treebank. *In Proceedings of the Workshop on Machine Translation and Parsing in Indian Languages* (pp. 53-60).

Borah, Pranjal Protim, Gitimoni Talukdar, and Arup Baruah. 2019 WSD for Assamese Language. *In Recent Developments in Machine Learning and Data Analytics*, pp. 119-128. Springer, Singapore

Himdweep Walia, Ajay Rana, Vineet Kansal. 2017. A Naïve Bayes Approach for working on Gurmukhi Word Sense Disambiguation", *6th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions), Amity University Uttar Pradesh, Noida, India*

Ide, Nancy & Jean Véronis. 1998. Word sense disambiguation: The state of the art. Computational Linguistics

Khaled Abdalgader M. OmarAndrew Skabar. 2012 Sense disambiguation using context vectors and sentential word importance," ACM Transactions on Speech and Language Processing, vol. 9, no. 1, pp. 1-21

Muhammad Abid, Asad Habib, Jawad Ashraf, and Abdul Shahid. 2017. Urdu word sense disambiguation using machine learning approach". Cluster Computing, pages 1—8

Nancy Ide and Jean Veronis 1998, Introduction to the Special Issue on Word Sense Disambiguation: The State of the Art

Nazima Mehdi and Aadil Ahmad Lawaye 2011. Development of Unicode Complaint Kashmiri Font: Issues and Resolution. Interdisciplinary Journal of Linguistics (IJL), 4,195-200

Pandit, Rajat, and Sudip Kumar Naskar. 2015. A memory based approach to word sense disambiguation in Bengali using k-NN method. *2015 IEEE 2nd International Conference on Recent Trends in Information Systems (ReTIS),* Kolkata, pp. 383-386.

Singh, Richard Laishram, Krishnendu Ghosh, Kishorjit Nongmeikapam, and Sivaji Bandyopadhyay 2014. A Decision Tree Based Word Sense Disambiguation System In Manipuri Language. Advanced Computing: An International Journal 5(4) p 17.

Tarjni Vyas, Amit Ganatra 2019. Gujarati Language Model:Word Sense Disambiguation using Supervised Technique. International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8, Issue-2S11, September 2019

Sentimental Poetry Generation

Kasper Aalberg Røstvold and **Björn Gambäck**
Department of Computer Science
Norwegian University of Science and Technology
Trondheim, Norway
kasperaarr@gmail.com, gamback@ntnu.no

Abstract

The paper investigates how well poetry can be generated to contain a specific sentiment, and whether readers of the poetry experience the intended sentiment. The poetry generator consists of a bi-directional Long Short-Term Memory (LSTM) model, combined with rhyme pair generation, rule-based word prediction methods, and tree search for extending generation possibilities. The LSTM network was trained on a set of English poetry written and published by users on a public website. Human judges evaluated poems generated by the system, both with a positive and negative sentiment. The results indicate that while there are some weaknesses in the system compared to other state-of-the-art solutions, it is fully capable of generating poetry with an inherent sentiment that is perceived by readers.

1 Introduction

Poetry generation is a type of linguistic creativity that requires certain qualities in both form and content, as well as the creation of understandable, meaningful and poetic language. A central part of poetry is the experience of the reader, including the emotions poetry can evoke. The overarching goal of this work is to explore and develop methods for generating poetry with a specific (pre-defined) inherent sentiment, which can be experienced by the readers. Earlier approaches to poetry generation followed a range of paths, such as methods based on templates (Gonçalo Oliveira, 2012) or corpora (Colton et al., 2012), evolutionary (Levy, 2001) or Case-Based Reasoning (Gervás, 2001) approaches, and generate-and-test (Gervás, 2000) or Blackboard (Misztal and Indurkhya, 2014) architectures. However, in recent years deep learners have proven powerful as poetry generators, including systems combining neural models with other techniques. As described below, Long Short-Term Memory (LSTM, Hochreiter and Schmidhuber, 1997) networks are the most used solutions in state-of-the-art systems, so an LSTM was implemented here, with an experimental focus on which specific network architecture and parameter settings would produce the best poetry word prediction model. The sentiment of the poems to be generated was decided in advance, with human judges rating their quality and how well the sentiment was perceived.

2 Related Work

Zhang and Lapata (2014) used a Recurrent Neural Network (RNN) to generate Chinese quatrains (stanzas with four lines), with the first line based on user-provided keywords giving the main concepts of the poem. Subsequent lines were generated based on previous lines, subject to admissible tonal pattern and structural constraints. Yi et al. (2016) also generated Chinese quatrains line-by-line based on user keywords, but using a sequence-to-sequence model with attention mechanism (Bahdanau et al., 2014), with a bi-directional RNN with gated recurrent units (GRU; Cho et al., 2014) as encoder-decoder to learn semantic relevance. Wang et al. (2016a) used a similar approach for character-by-character iambics generation, utilising a bi-LSTM as encoder and another LSTM as decoder to alleviate the quick-forgetting problem associated with conventional RNNs.

Ghazvininejad et al. (2017) combined hard format constraints with an RNN to generate 14-line classical sonnets in iambic pentameter, given a user-supplied topic and a set of related words, using *word2vec* (Mikolov et al., 2013). Rhyme words were found using CMU Pronouncing Dictionary (CMUdict),[1] with fixed pairs of often used words added to make the system find rhymes in rare topics. A Finite-state acceptor was built with paths for all

[1] http://www.speech.cs.cmu.edu/cgi-bin/cmudict

Proceedings of the 17th International Conference on Natural Language Processing, pages 246–256
Patna, India, December 18 - 21, 2020.

conceivable vocabulary word sequences obeying formal rhythm constraints, and an RNN selected the best path. Similarly, Benhart et al. (2018) combined an RNN with sonnet format constraints, but added part-of-speech restrictions to alleviate erroneous word choices, and dynamically trainable word embeddings, so that the language model was able to learn some grammar before adjusting its word representations to suit the training corpus.

Wang et al. (2016b) implemented a 2-phase system: planning and generation, with an encoder-decoder generator using bi-GRUs. A special planning schema in advance selected sub-keywords (based on user input) guided by a language model and each line then generated with the planned word to improve coherence. Zhang et al. (2017) used the same attention-based RNN generation model, but with an additional memory component, effectively giving regularisation that constrains and modifies the behaviour of the model. Van de Cruys (2020) also utilised an encoder-decoder, but generated poems in both English and French. Human evaluators scored the output highly with regard to fluency, coherence, meaningfulness and poeticness, even though only non-poetic text was used as training data for the generator.

Unlike the one-pass generation for previous neural networks models, Yan (2016) proposed a generative model with a polishing schema, refining the RNN-generated poem through several iterations. Also, while previous models were based on maximum likelihood estimation (MLE), which optimizes word-level loss and can lead to the systems remembering common patterns of the training corpus, Yi et al. (2018) added reinforcement learning to a basic generator pre-trained with MLE, simultaneously training two generators that learn both from the teacher (rewarder) and from each other. Automatic rewarders were designed corresponding to four criteria: fluency, coherence, meaningfulness, and overall quality.

Tikhonov and Yamshchikov (2018b) aimed to generate poetry in the style of a specific author, using an LSTM to predict the next word based on a previous word sequence, with the embeddings of the document currently being analysed used to support the model at every step. Two datasets were used to train the model, in English and Russian.

Several of the systems presented above implement a form of user input to influence the mood of the poetry, often related to a given sentiment. However, two such systems are particularly important in the way they include the use of sentiment: In the corpus-based approach of Full-FACE Poetry Generation (Colton et al., 2012), the system decides on a mood by checking the average sentiment of a set of newspaper articles posted during the previous 24 hours, and then selects one of the five articles with the highest resp. lowest sentiment value. Misztal and Indurkhya's (2014) system includes an emotional personality aspect implementing both sentiment analysis and emotional modelling. To extract sentiment, the Sentistrength (Thelwall et al., 2010) tool is used, rating positive and negative scores on a *valence* value scale. An average *arousal* value of the input is calculated using Affective Norms for English Words (ANEW, Bradley and Lang, 1999), and the poem's emotional state is set by combining valence and arousal. WordNet-Affect (Strapparava and Valitutti, 2004) is used to build a hierarchy of words describing emotional states in order to generate the affective content of poems.

3 Data set

The data set used in the experiments here is the English part of the data collected by Tikhonov and Yamshchikov (2018a,b). It consists of poems written and published by users on a public website, which leads to a variance in the quality of content, but both the large size and variance in content are positive factors for network training. The provided data set was already cleaned, with all types of punctuation removed and all letters converted to lowercase. However, there were some inconsistencies in how contractions were represented, with some appearing in a joined form (e.g., *wouldve*) and others as separate words (*would ve*). Hence all spaces between regular contractions were removed, as were line break markers (`<br>`), with every individual poem was instead represented by single individual lines, so that the structure of the generated poems would not be constrained.

The original data set contains $3,943,982$ poems and $155,066,504$ tokens, with a vocabulary of $708,727$ unique tokens. Most of the unique tokens come from misspellings, alternative spellings, irregular words and names. The training data was shortened to specifically reduce the vocabulary size, in order to remove words that very rarely appear in the data set, and to reduce the size of the matrix representation of data used in training the network. To reduce the data set, tests were run on vocabu-

	Lemmas	Poems	Tokens	Polar
Data set 1	10 000	205 230	15 847 356	1 849
Data set 2	15 000	306 942	25 883 608	2 424
Data set 3	20 000	395 057	35 178 076	2 900
Data set 4	30 000	530 121	50 505 342	3 519

Table 1: Data set sizes based on vocabulary

	Lemmas	Poems	Tokens
Training set 1	9 195	12 273	1 300 068
Training set 2	18 031	26 873	3 106 347
Test set 1	9 195	4 620	405 286
Test set 2	18 031	4 307	406 324

Table 2: Training and test data sets

lary sizes of $10,000 - 30,000$, finding how many of the poems only included words within a given vocabulary, as can be seen in Table 1. Since larger vocabulary results in that more possible words can be generated, but negatively effects training time and is reliant on how much the hardware used for training can handle, a simple test was run on a GeForce GTX 1070 graphics card. Creating a bidirectional LSTM using Keras[2] with a single hidden layer of $1,024$ neurons, and training on a random sample size of the $5,000,000$ tokens, with different vocabulary sizes, resulted in the GPU experiencing memory problems when exceeding a vocabulary size of $30,000$, so the vocabulary size was capped at that value, while the lower limit was set to $10,000$, as smaller vocabularies would result in too few words available for generation.

An important aspect of the vocabulary is the inclusion of words with sentiment values, since they would be generated to add sentiment to the poetry. Using Vader (Hutto and Gilbert, 2015), the data sets were investigated for how many unique words they contained with a polar (non-neutral) sentiment, i.e., either having a positive or a negative sentiment. Those are also reported in Table 1.

Due to the increased number of unique sentiment words with increased vocabulary sizes, but also due to memory restrictions and larger vocabularies resulting in greater training times, the vocabulary sizes for neural network training was chosen to be $10,000$ and $20,000$. As Table 1 shows, the number of individual tokens in the data set with a vocabulary size of $20,000$ is over 35 million, which is too large to train on, concerning the time it would take. The data sets were therefore further reduced to *training data sets*, containing only 10% of the original data. All poems were ordered after the user name of the person that published it, so every tenth poem was selected for creating training data, to get poems from as many different authors as possible. The training data set sizes are presented in Table 2.

To evaluate the trained networks, new *test* data was created from the original data sets, in the same way as the training data, but from poems not included in the training data and only containing tokens included in the training data vocabularies. The test data sets are also shown in Table 2.

The training of the neural network is done by creating input sequences to be fed through the network, but also creating the correct output which is then compared to the output of the network. The input is therefore created by choosing a sequence of the training data with a given length, and the token following that sequence as the correct output. A training data sequence length of 5 was chosen, both for memory storage reasons and since 5 was decided to be the shortest possible line length of the generated poetry. Resticting the sequence length will also make the network predict the next words based on just a short sequence, instead of all of the poem that is already generated, which could lead to more variation. Before creating the network input matrices, every poem was reversed, with the last word being the first, and so on, following the approach used by Benhart et al. (2018). This is done so it is possible to start with the ending rhyme word of a line of poetry, and generate the rest of the line backwards from that rhyme-word.

4 Architecture

This section introduces the architecture for the system implemented in this project. The first part describes the long short-term memory network that was implemented, while the second part describes the complete poetry generation process.

4.1 The LSTM network

A bi-directional Long Short-Term Memory neural network is the main component in the poetry generation process. After being trained on a large data set of human-written poetry, its task is to give a prediction on the next word that should follow after a given input sequence of words. The prediction consists of an array, with a predicted score of every unique word in the vocabulary.

Figure 1 shows the Bi-LSTM network. The input with a sequence length of 5 is transformed into

[2]https://keras.io/

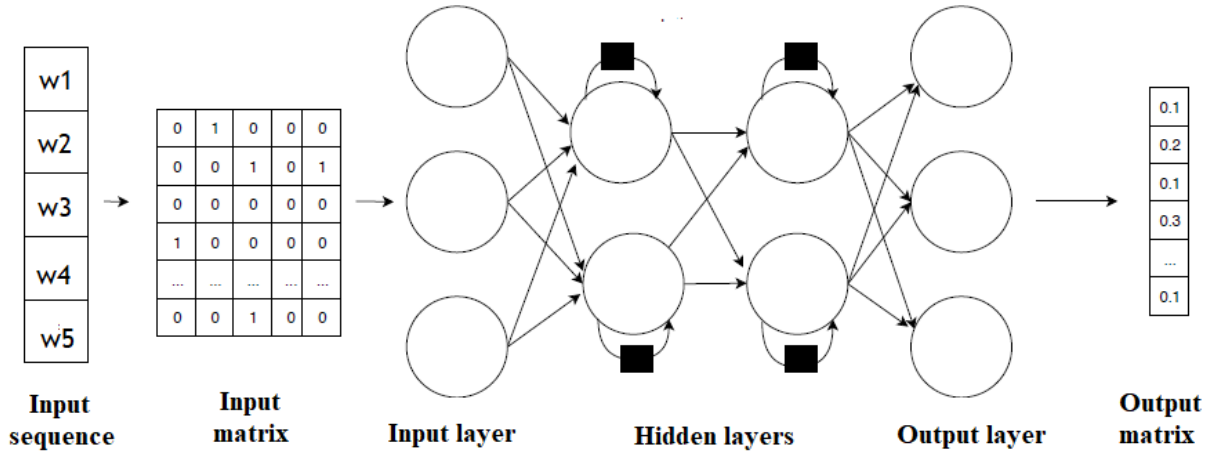


Figure 1: LSTM network prediction process

a matrix representation before being fed to the network. The network's input layer represents the matrix data given to the network, which accounts for both the data used for training and for the input during the poetry generation process. The output layer is a softmax activation function layer that outputs a matrix representing the network predictions for all possible words. During training, the result of these predictions are compared to the actual following word of a given sequence, which will update the hidden layer(s) based on a loss function, which the network tries to minimise. Between the input and output layers are hidden layers, consisting of (recurrent) LSTM cells that compute the possible values for the next predictions, based on the current input and the (stored memory of) the previous part of the input sequence.

During training, the network tries to minimise the cross entropy loss:

$$L = -\frac{1}{N}\sum_{c=1}^{N} ln(p_c) \quad (1)$$

where N is the total training set, p the prediction, and c the category (the word) being looked at. The loss is the cross entropy between the distribution of the true labels and the network predictions. To minimise the loss function, backpropagation is used to update the network weights during training, by taking the error found by the loss function L and calculating the gradient of L with respect to the weights, w, in the network, $\frac{\partial L}{\partial w}$. The gradient is fed to the optimiser, which updates the weights in an attempt to minimise the loss function. The optimiser used is stochastic gradient descent, a first-order iterative optimisation algorithm. It is possible to get stuck in a local minimum when minimising the loss function, therefore the learning rate is initially set higher, and decreases during the training to try to find the global minimum. In addition, dropout (Srivastava et al., 2014) was used during network training to reduce overfitting.

4.2 Poetry generation system

The network output consist of an array containing the predicted value for each word in the vocabulary, to follow the input word sequence fed to the network. In addition to the LSTM, the poetry generator has three important components: rhyme pair generation, prediction score updating, and tree search. The generation of rhyme pairs is used as initial input for generating each poetry line and ensures that the poetry contains end rhymes. The score updating algorithm adjusts the prediction values gained from the LSTM, by adding rules and different weightings on certain types of possible words, to enhance the quality of the generated sequences. The search tree expands the number of possible sequences created during the generation, increasing the chance of finding the best possible sequence to create a poem from.

Rhyme word generation: The first part of the generation process consists of finding rhyme word pairs, that are used as input for producing a poetry line, as it is generated backwards from the rhyme words. For this, a unique word having a sentiment value matching the decided sentiment is randomly chosen from the vocabulary, using Vader (Hutto and Gilbert, 2015), with words with a sentiment value above 0.0 selected if the sentiment is to be positive, and less than 0.0 for negative.

The vocabulary is then searched for another word of the same sentiment rhyming with the first

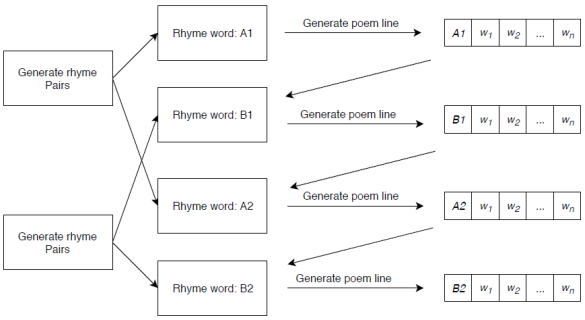


Figure 2: Poetry generation with rhyme word input

word, with CMUdict identifying the syllables. The conditions that need to be met to complete a rhyme do not form a *perfect* rhyme (i.e., with identical stressed vowel sounds in both words, as well as any subsequent sounds, but with different consonant preceding the stressed vowel), but rather a form of *imperfect* rhyme, where the last three sounds, including a consonant, are equal for two words. Each word in a rhyme pair is used for one line of poetry, and the rhyme form chosen is *ABAB*, so two pairs are needed to generate one 4-line stanza.

The first rhyme word found is used to predict the following words for the first line (which will be the final line of the poem, since it is generated backwards). The last four words of the generated sequence, plus the rhyme word for the next line, are used as input to generate the next line. This process is presented in Figure 2, where the rhyme word A1 is used to generate the first line, consisting of the rhyme word and a sequence of n words w. The next rhyme word B1 is then added to the sequence of the four words from w_{n-3} to w_n from the last sequence, and used as input to generate the next line. The last generated line with the rhyme word B2 is the first line of the complete poem.

Updating prediction scores: For every word generated in a sequence, predictions on all the possible words are given by the LSTM, based on network input consisting of the previously generated words. Four algorithms were implemented for updating and adjusting the prediction scores, to improve the generated poetry. These algorithms were similarly used in other state-of-the-art-solutions, including Benhart et al. (2018) implementing repetition and sentence structure restrictions, and Colton et al. (2012) measuring sentiment values of poetry lines against a set value.

(i) Since some popular words often appear in the data set, they will have a high prediction value. To avoid a repetitive use of these words and to achieve better variety, **related words** are found using WordNet (Miller, 1995) for the 20 unique words with the highest prediction value, and the predictions for those words are increased, thus increasing the probability of choosing a less used word.

(ii) To reduce the likelihood of a line of poetry containing **repeated words**, a word's predicted score is reduced during sequence generation, if it has previously appeared in the sequence.

(iii) It was observed that the generated poetry consistently had obvious part-of-speech (POS) errors. Sequences were thus POS-tagged using the Natural Language ToolKit (Bird et al., 2009), and used to implement **sentence structure** restrictions, such as disallowing a pronoun directly preceding another pronoun and a verb directly preceding another verb.

(iv) Based on the intended **sentiment** for the poems, the scores for the possible words with a corresponding sentiment were increased, using Vader to find all the possible next words in a sequence having a sentiment value corresponding to the intended sentiment and increasing their predicted score, resulting in an increased chance of choosing words with the correct sentiment value when generating sequences. The degree of increasing predicted scores is based of the sentiment value of each word. Words that Vader evaluates as having a higher correct sentiment value get a larger predicted score increase; words with the opposite sentiment get their predicted scores decreased.

Search tree algorithm: To increase the chances of finding the best possible sequences to create a poem from, a tree search algorithm was implemented to expand the number of sequences created. Instead of generating a single next word based on the highest score, the search takes a number of the possible words with the highest score values, and generates different possible sequences. This is repeated for every new word in the sequence.

The first word w_1 can be one of the rhyme words used to generate the rest of the sequence, but it can also be a sequence of previously generated words plus the new rhyme word. Using this as the input to the LSTM network, we get the predictions for the next words. These prediction scores are then updated by the four algorithms described above, before 20 new sequences are created, that all consist of the first word w_1 plus one of the next words

x with one of the highest scores, where every sequence has a unique next word x. This generation step is then performed on all the possible sequences created in the previous step, for adding the third word in the sequence, and so on.

The search continues with a number of steps i equal to the total number of words to be generated in a given sequence. As the search tree grows exponentially, the possible sequences that are created with the lowest *scores* are continuously pruned throughout the process.

When all the possible sequences have been created, the sentiment values for each complete sequence are evaluated using Vader. These sentiment values are again used to adjust the score for all the possible sequences. After this final score update, the sequence with the highest score is chosen, and used to generate a new line of the current poem.

The process described above is used to generate 8-line poems consisting of two 4-line stanzas with a rhyming scheme of *ABAB CDCD*. The system adds commas after each of the first three lines of a stanza, a period after the last line, and an empty line between the two stanzas. The first letter of the first word of every line is capitalised, in addition to other letters where capitalising is grammatically correct. Finally, Vader is used to calculate the sentiment value of the entire complete poem. This final sentiment value is used in the experiments where it is compared to human evaluations.

5 Network Experiments

Experiment were conducted to decide the final architecture of the LSTM network, and what parameters to use in the implementation of the poetry generation system. The network training was performed on a Tesla P100-PCIE-16GB GPU. The learning rate was initially set to 0.9 for the training of all networks. A monitor was implemented on the validation loss calculated on validation samples each epoch, reducing the learning rate after a given period when the validation loss has not decreased. The period before reducing the learning rate was set to 5 epochs, and the reduction of the learning rate to a factor of 0.3. The minimum limit for the learning rate was set to 0.001. The dropout probability rate was 0.6 for all non-recurrent units. This value was chosen based on Zaremba et al. (2014), where the dropout rate used for medium LSTM (650 units per layer) was 0.5, and 0.65 for large LSTM (1500 units per layer).

	M1	M2	M3	M4	M5
CA	2.25	2.38	2.46	2.41	2.33
L	8.997	8.097	7.951	8.131	8.813
WP	8,077	3,286	2,839	3,400	6,722

Table 3: Results using Training Data set 1

Two different data sets were used in the experiments and the parameters tested were the numbers of hidden layers, units in each hidden layer, and training epochs. The test data input was fed through the networks, and predictions were measured against true values, using three measures:

(i) *Categorical accuracy* (CA) is calculated for the entire test data set, by taking the mean accuracy rate across all the predictions, to check if the predicted word is equal to the true word.

(ii) *Categorical cross entropy loss* gives the loss function (L; Eq. 1) used during network training.

(iii) *Word perplexity* (WP) measures how well a probability distribution can predict a sample: the lower the perplexity, the less confused a network is about predicting the next word. It is calculated by using the loss function as exponent to the power of the constant e:

$$\text{WP} = e^{L} = exp(-\frac{1}{N}\sum_{c=1}^{N} ln(p_c)) \qquad (2)$$

5.1 LSTM network training

Five LSTM networks were trained over 25, 50 or 75 training epochs on each of the training data sets: two network models with two hidden layers, with 256 (denoted as model M1 below) resp. 1024 (M2) hidden units; and three models with three hidden layers, with the number of hidden units being 256 (M3), 512 (M4) or 1024 (M5).

The LSTM models trained on Training set 1 (with a vocabulary size of $9,195$) were only evaluated on the Test set 1, since it has the same size. Table 3 displays the categorical accuracy, cross entropy loss, and perplexity for those networks.

The LSTM network models trained on the Training set 2 on the other hand (with a vocabulary size of $18,031$), were evaluated on both Test set 1 and Test set 2, in order to be able to compare LSTM models trained on different vocabulary sizes, with results presented in Table 4.

	M1	M2	M3	M4	M5
Evaluation Data set 1 results					
CA	2.53	2.38	2.51	2.41	2.25
L	7.986	8.617	8.000	8.315	9.729
WP	2,939	5,522	2,981	4,084	16,789
Evaluation Data set 2 results					
CA	2.34	2.24	2.38	2.24	2.07
L	8.266	8.893	8.617	8.540	9.983
WP	3,888	7,279	5,522	5,117	21,653

Table 4: Results using Training Data set 2

5.2 LSTM network evaluation

The perplexity varied greatly for the different LSTMs, ranging from $2,939$ to $21,653$, and increasing with increased network complexity. However, networks trained on a smaller vocabulary did not have better perplexity scores than those trained on the larger vocabulary, so the ones trained on the smaller vocabulary were disregarded, since a smaller vocabulary means fewer possible unique words in the generated poems.

The network trained on the large vocabulary with best perplexity ($3,888$ for Test set 2) had 2 hidden layers and 256 hidden units per layer. However, generating text solely based on predictions given by this network, it showed signs of being highly overfitted, repeating a few select words. Since this network did not achieve perplexity scores significantly better than the alternatives, the larger network with the next-best perplexity was chosen instead. It has 3 hidden layers, 512 hidden units per layer, and was trained for 25 epochs, achieving a perplexity of $5,117$ for Test set 2, with a cross entropy loss of 8.540 and a categorical accuracy of 0.0224. A network with the same architecture, but trained for an additional 25 epochs, resulted in a much higher word perplexity score ($21,653$), likely due to underfitting the training data, since it had a much harder time predicting correct words.

While the perplexity differed greatly for the LSTMs, it was consistently very high, representing poor network training results. Zaremba et al. (2014) achieved a word perplexity of 78.4 with a regularised LSTM where dropout was used, training on the Penn Tree Bank (PTB) 10k vocabulary (Marcus et al., 1993). While the vocabulary size of the training data does not differ, there are several notable differences that will impact the results: Word perplexity will be greater for data with a larger vocabulary size (plainly due to the word possibilities increasing, clearly shown in Table 4. Furthermore, the PTB consists of grammatically correct literature, while the data used here consists of publicly written poetry, which is more irregular and with greater variation (especially in sentence structure and grammaticality), which could impact the networks' ability to learn patterns from the texts. The networks here were also trained on short sequences with a length of 5, making it harder for them to learn connections and general rules in the data.

	Positive sentiment	Negative sentiment
Rated positive	58.8%	2.7%
Rated neutral	36.2%	27.6%
Rated negative	5.0%	69.7%
Human average	0.360	-0.372
Vader average	0.977	-0.925

Table 5: Sentiment evaluation results

6 Evaluating the generated poetry

Human judges evaluated the generated poetry both in itself and with regard to the intended sentiment. The poetry was rated along the dimensions suggested by Manurung (2004): Grammaticality, Meaningfulness, and Poeticness, on a 1–3 scale (1 being "not", 2 "partially, and 3 "fully"). For the sentiment rating, the human judges evaluated the poetry using three categories: Negative sentiment, No sentiment (neutral), and Positive sentiment. If a poem was evaluated as having negative or positive sentiment, it was graded with a score of 1–3 ("Slightly", "Quite", and "Very" negative/positive).

Twenty poems were evaluated (i.e., 40 stanzas and 160 lines), generated to contain 10 each with positive and negative sentiment. Thirty human judges participated, evaluating a selection of 6 or 7 poems each, with a near-equal amount of positive and negative sentiments.

The 20 poems were scored with an average mean of Grammaticality: 1.488 ± 0.0476, Meaningfulness: 1.582 ± 0.0338, and Poeticness: 2.012 ± 0.0342. Table 5 shows the percent of the judges who rated the poems to contain positive, neutral or negative sentiment. The evaluators also rated the degree of perceived sentiment for each poem they had evaluated to contain a positive or negative sentiment, with ratings for every poem normalised to values between 0–1. The poems included in the experiments were also rated using Vader. Table 5 also shows the average sentiment degree scores.

The results show that on average the majority of judges perceived the poems to contain the sentiment value intended by the system, but the degrees of the sentiment value, both for negative and positive poems, are rated considerably lower by the human judges than the ratings given by Vader. One reason for this is the very high ratings that Vader assigns, both for negative and positive sentiment scores, with the generator discouraging the use of words with an opposite sentiment value (compared to the intended value), while encouraging the use of words with a "correct" sentiment value, given by Vader. The degree ratings from the human judges on the other hand reflect that a lack of the words with an opposite sentiment value does not result in a high degree of sentiment being perceived. It is also interesting to note the consistency of the degree ratings given by the human judges, where positive and negative sentiment poems are on average rated with almost the same degree of sentiment.

Three of the generated poems are presented below: #8 with one of the lowest scores in the experiment results, and #3 and #14 with some of the highest. Along with each poem are its scores, with the first array presenting the grammaticality, meaningfulness and poeticness scores. The second array shows the percentage of human judges who found the poem to contain negative, neutral, or positive sentiment. Poem 8 achieved a score of 1.22 for both grammaticality and meaningfulness, while Poem 14 achieved a score of 1.73 for those. The poeticness scores for both Poem 8 and 14 are below average, while Poem 3 achieved the highest poeticness score of all the poems with 2.5, and also high scores for grammaticality and meaningfulness.

A common trait among all the generated poems is incorrect use of articles and conjunctions, in addition to erroneous use of other word classes and poor sentence structure. Examples of this are the sequences *So most from an till* and *Amid to our so most from currently* in Poem 8. Another noticeable aspect is the rhyme pairs in the poetry not always rhyming, e.g., *exceptions* and *solutions* at the end of lines 6 and 8 in Poem 14. This is due to the system pairing rhyme words based on the last three syllables using CMUdict. A similar aspect is the consistent lack of repetitiveness found in the poetry, since repeated use of words is highly discouraged by the system, to avoid constant use of the same high predictions words, resulting in the poems lacking an often used poetic technique, to consciously

Poem 3

Of ravishing sin was naturally deprivation, 1
Of war suffering it tired than two then a situation, 2
And the go on no lies at finding they frustration, 3
Of voice are their seven then limitation. 4

The forest in mystic hands understood, 5
A trapped on must words most from your unarmed, 6
With wind raging to our misunderstood, 7
As a entire my that alarmed. 8

Metric evaluation scores: [1.7, 1.5, 2.5]
Sentiment evaluation scores: [0%, 20%, 80%]

Poem 8

For worst the all sleep as hearted of unpredictable, 1
So most from an till sensations, 2
Poor tall being eye in goes horrible, 3
Without the when todays so most from contradictions. 4

Kept on humanitys your soft in night a frightening, 5
Amid to our so most from currently, 6
In unwanted of feminine with sickening, 7
By no lies at impatiently. 8

Metric evaluation scores: [1.22, 1.22, 1.89]
Sentiment evaluation scores: [0%, 56%, 44%]

Poem 14

Without the oozing in night quickly divine thin lovable, 1
Like non focused best gods so wearing on contentment, 2
Bliss most from victory like favorable, 3
Without the respected of your improvement. 4

A dusty amid to governments, 5
Like mother of god be will certain the exceptions, 6
Amid to our so most from the do innocents, 7
In love a feeling most which are their solutions. 8

Metric evaluation scores: [1.73, 1.73, 1.91]
Sentiment evaluation scores: [82%, 18%, 0%]

repeat words or phrases. Other noticeable factors include misspellings (e.g., *humanitys* in Poem 8), and use of rare and special words. The generator vocabulary consists of the $20,000$ most frequently used words in the chosen data set, so any spelling error would mean that a high frequency of that misspelling occurs in the data.

The average scores for both grammaticality and meaningfulness are rather low. This correlates directly with the results from the LSTM training: the poor word perplexity scores for the LSTM networks have an effect on the words chosen by the poetry generator, creating poems having poor grammaticality, and therefore being more difficult to perceive meaningfulness from. The variables that update the word predictions during the generation could be improved to positively impact the evaluation scores, in particular the feature that updates prediction values based on sentence structure.

Poeticness had a significantly higher score, which could be due to factors not influenced by

the LSTM model's performance, specifically including the generation and use of rhyming pairs and the form of the poetry. The form of the poetry is also not influenced by the LSTM predictions, as the length of each line is randomly decided between two outer bounds, and proper punctuation is added after each line, including a line break between stanzas. While consistent rhyme form, punctuation, and varying line lengths likely account for the good poeticness results, possible weaknesses that might have affected the results are the lack of perfect rhymes and the lack of known poetry forms with consistent syllable lengths, such as sonnets. The lack of repetitiveness as encouraged by the system might also negatively affect the score.

7 Conclusion and Future Work

A system capable of generating poetry with inherent sentiment has been designed and implemented, with the main system component being a bi-directional Long Short-Term Memory network used for generating word predictions based on a given input sequence. The network was trained on a data set consisting of poetry written by humans. Other components of the final generation system are algorithms and rule-based methods for influencing word predictions and word choices during the generation process, and a search algorithm for expanding the possibilities of generated sequences.

The implemented system was used to generate 20 poems in total, all consisting of two stanzas with four lines each. 10 of the poems were generated to contain an inherent positive sentiment value, while the other 10 were generated to contain a negative sentiment value. Several experiments were conducted, both regarding the LSTM network and on the generated poetry. The first experiment was on training different LSTM networks with varying architecture details, with the goal of training the best performing network model to use in the implementation of the final poetry generation system.

Two other experiments were conducted on the final generated poetry, both involving human judges evaluating the generated poetry. The first of these experiment consisted of the judges evaluating the poetry based on three standard evaluation criteria. This enabled evaluation of the performance of the poetry generation system, and comparison to other works that have been conducted in this field. In the second experiment the human judges evaluated the sentiment value they perceived generated poetry to contain, in order to investigate whether the system was capable of generating poetry with an inherent sentiment value that would be perceived as intended by human readers.

The results of the experiments varied, with LSTM experiments giving word perplexity scores worse than state-of-the-art solutions. Applying some standard evaluation metrics showed one of the metrics achieving similar values to state-of-the-art solutions, while the other metric gave poorer results, one reason being the influence from the subpar prediction performance of the LSTM network. The experiment for evaluating the sentiment of the generated poetry produced good results. While there is a lack of similar experiments by others to compare to, the results show a clear trend of the human judges perceiving the poetry to contain the intended sentiment value.

Possible future work could include implementing additional features or other architectures, such as word embedding models or language models like BERT (Devlin et al., 2018) that show prominent results for text analysis, or use mutual reinforcement, which has given state-of-the-art results in poetry generation (Yi et al., 2018). The data set used to train the neural network model has a considerable effect on the system's performance and the generated poetry. Hence using different data sets, especially data containing poetry of a generally accepted higher quality, would probably improve the system. Adding only perfect rhymes for rhyme pair generation, or a strict poetic form based on syllables, such as the sonnet form, could improve the poetic qualities of the output.

The main feature of the generation system is to generate poetry with an inherent sentiment, and this can also be further developed. First, the system needs to generate poetry with a wider range of sentiment value words, as it current only uses words with neutral sentiment or sentiment values corresponding to the intended sentiment. Adding more words with opposite sentiment could increase the poetry's emotional dynamic. The sentiment feature could also be extended to generate poetry with a wider range of emotions, e.g., by using emotional modelling similarly to Misztal and Indurkhya (2014), or by adapting the system to generate poetry with an inherent degree of a specific sentiment, not just a general negative or positive value.

Acknowledgments

Thanks to Trond Aalberg and the AI Master students at NTNU's Department of Computer Science for discussions, insights and helpful feedback.

Thanks also to the people who participated in the experiments that were conducted and to Tikhonov and Yamshchikov for providing the data set used.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

John Benhart, Tianlin Duan, Peter Hase, Liuyi Zhu, and Cynthia Rudin. 2018. Shall I compare thee to a machine-written sonnet? An approach to algorithmic sonnet generation. *CoRR*, abs/1811.05067.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Margaret M. Bradley and Peter J. Lang. 1999. Affective norms for English words (ANEW): Instruction manual and affective ratings. Technical Report C-1, Center for Research in Psychophysiology, University of Florida, Gainesville, FL, USA.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Simon Colton, Jacob Goodwin, and Tony Veale. 2012. Full-FACE poetry generation. In *Proceedings of the 3rd International Conference on Computational Creativity*, pages 95–102, Dublin, Ireland.

Tim Van de Cruys. 2020. Automatic poetry generation from prosaic text. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2471–2480, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Pablo Gervás. 2000. WASP: Evaluation of different strategies for the automatic generation of Spanish verse. In *Proceedings of the AISB'00 Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science*, pages 93–100, University of Birmingham, England.

Pablo Gervás. 2001. An Expert System for the Composition of Formal Spanish Poetry. In *Applications and Innovations in Intelligent Systems VIII*, pages 19–32, London, England. Springer.

Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. 2017. Hafez: an interactive poetry generation system. In *Proceedings of ACL 2017, System Demonstrations*, pages 43–48, Vancouver, BC, Canada. Association for Computational Linguistics.

Hugo Gonçalo Oliveira. 2012. PoeTryMe: A versatile platform for poetry generation. In *Proceedings of the Workshop on Computational Creativity, Concept Invention, and General Intelligence*, pages 21–26, Montpellier, France. Publications of the Institute of Cognitive Science.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

C.J. Hutto and Eric Gilbert. 2015. VADER: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the 8th International Conference on Weblogs and Social Media, ICWSM 2014*, pages 82–91, Ann Arbor, MI, USA. AAAI Press.

Robert P. Levy. 2001. A computational model of poetic creativity with neural network as measure of adaptive fitness. In *Proceedings of the Workshop on Creative Systems, International Conference on Case-Based Reasoning*, Vancouver, BC, Canada.

Hisar Maruli Manurung. 2004. *An evolutionary algorithm approach to poetry generation*. Ph.D. thesis, Institute for Communicating and Collaborative Systems, School of Informatics, College of Science and Engineering, University of Edinburgh, Edinburgh, Scotland.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

George A. Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

Joanna Misztal and Bipin Indurkhya. 2014. Poetry generation system with an emotional personality. In *Proceedings of the 5th International Conference on Computational Creativity*, pages 72–81, Ljubljana, Slovenia.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Carlo Strapparava and Alessandro Valitutti. 2004. WordNet-Affect: an affective extension of WordNet. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 1083–1086, Lisbon, Portugal.

Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. 2010. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, 61(12):2544–2558.

Aleksey Tikhonov and Ivan Yamshchikov. 2018a. Sounds Wilde. phonetically extended embeddings for author-stylized poetry generation. In *Proceedings of the 15th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 117–124, Brussels, Belgium. Association for Computational Linguistics.

Alexey Tikhonov and Ivan P. Yamshchikov. 2018b. Guess who? Multilingual approach for the automated generation of author-stylized poetry. In *2018 IEEE Spoken Language Technology Workshop*, pages 787–794, Athens, Greece. IEEE.

Qixin Wang, Tianyi Luo, Dong Wang, and Chao Xing. 2016a. Chinese song iambics generation with neural attention-based model. *CoRR*, abs/1604.06274.

Zhe Wang, Wei He, Hua Wu, Haiyang Wu, Wei Li, Haifeng Wang, and Enhong Chen. 2016b. Chinese poetry generation with planning based neural network. *CoRR*, abs/1610.09889.

Rui Yan. 2016. I, Poet: Automatic poetry composition through recurrent neural networks with iterative polishing schema. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2238–2244. AAAI Press.

Xiaoyuan Yi, Ruoyu Li, and Maosong Sun. 2016. Generating Chinese classical poems with RNN encoder-decoder. *CoRR*, abs/1604.01537.

Xiaoyuan Yi, Maosong Sun, Ruoyu Li, and Wenhao Li. 2018. Automatic Poetry Generation with Mutual Reinforcement Learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3143–3153, Brussels, Belgium. Association for Computational Linguistics.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR*, abs/1409.2329.

Jiyuan Zhang, Yang Feng, Dong Wang, Yang Wang, Andrew Abel, Shiyue Zhang, and Andi Zhang. 2017. Flexible and creative Chinese poetry generation using neural memory. *CoRR*, abs/1705.03773.

Xingxing Zhang and Mirella Lapata. 2014. Chinese Poetry Generation with Recurrent Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680, Doha, Qatar. Association for Computational Linguistics.

WEKA in Forensic Authorship Analysis: A corpus-based approach of Saudi Authors

Mashael M. AlAmr, Professor Eric Atwell
School of English and School of Computing
University of Leeds, Leeds, United Kingdom
{enmmaa, e.s.atwell}@leeds.ac.uk

Abstract

This is a pilot study that aims to explore the potential of using WEKA in forensic authorship analysis. It is a corpus-based research using data from Twitter collected from thirteen authors from Riyadh, Saudi Arabia. It examines the performance of unbalanced and balanced data sets using different classifiers and parameters of word grams. The findings further support previous studies in computational authorship identification.

1 Introduction

"Authorship attribution, broadly defined, is one of the oldest and one of the newest problems in information retrieval." (Juola, 2008, p. 287). It aims to identify or attribute one or more disputed texts to a single or multiple author(s), either from a closed set or an open one (Stamatatos et al., 1999; Koppel et al., 2009). Recent trends in forensic authorship analysis aim for incorporating artificial intelligence tools to find reliable results that are free of cognitive biases. WEKA (Witten et al., 2016) is a collection of machine learning algorithms that perform data mining tasks. Its tools can achieve pre-processing, classification, clustering, and capable of developing new machine learning schemes. Therefore, WEKA is ideal to pre-process, classify, and even create machine learning schemes for identifying authorship. This study aims to explore the potential of using WEKA in a corpus-based forensic authorship analysis research.

1.1 Research questions

This study proposes to answer the following questions:

1- What is the size of data required for WEKA to identify authorship accurately?
2- Which classifier can accurately identify authorship using the NASCT corpus?
3- Which parameter is most accurate to identify authorship using the NASCT corpus?

2 Related Literature

This section discusses the notion of idiolect and related literature of artificial intelligence in authorship analysis using short texts.

2.1 The term 'idiolect'

This term was coined by Bloch (1948), which is a blend of the Greek words 'idio' and 'lect' to better reflect the concept of *personal language variety*. He defines idiolect as an individual-level variety that consists of a uniquely patterned set of linguistic characteristics. The notion of examining the individual's production of language was dismissed till a later stage in language studies. Crystal (1997) stated that each individual has their own language system that generates their unique dialect. Turell (2010) highlighted the importance of the concepts of markedness and saliency as art of idiolectal style in forensic text comparison. A text is a distinctive production thereby making it marked as it conveys specific and accurate information of its producer. The concept of saliency in linguistics is connected to the idea of a prominent feature that can be easily noticed. The concept of saliency that works best in forensic text comparison is a combined approach of discourse analysis and corpus linguistics. An item or a feature is considered salient if it stands out statistically when comparing two subcorpora or when a subcorpus is compared to the totality of a corpus (Turell, 2010). For this study, the linguistics features examined are dialectal features of Najdi Arabic, a dialect spoken in the central region of Najd where Riyadh, the capital of Saudi Arabia is

Proceedings of the 17th International Conference on Natural Language Processing, pages 257–260
Patna, India, December 18 - 21, 2020.

located. They are 45 dialect-specific features classified into interrogatives, negatives, and deictic expressions (Alothman, 2012; Binturki, 2015).

2.2 AI in Authorship analysis

Studies in authorship analysis, authorship identification in particular, aim to find the optimum classifiers, parameters, and n-grams that achieve the task with the highest accuracy rates. Numerous studies confirm that accurate authorship identification results can be achieved using small sized data (Rico-Sulayes, 2011; Brocardo et al., 2014; Saha et al. 2018). Moreover, several studies found that Linear Support Vector Machine (SVM) demonstrate accurate classifying results compared to others. Decision Tree J-48 and Multinominal Naïve Bayes perform more accurately with numeric data (Brocado et al., 2014; Maruktat et al., 2014).

In terms of n-grams, character grams proved to perform well in short texts such as WhatsApp and Twitter but with some limitations to identify authors' texts without cross examination (Shrestha et al., 2017; Banga et al., 2018). As for word grams, some studies conclude that unigrams even in the shape of an emoticon can show good results in indetifying authorship (Fissette, 2010). Bigrams proved to be successful in identifying authorship in literary texts (Feiguina and Hirst, 2007).

In addition, body of literature has been published in the authorship identification field in Saudi Arabia that focuses on computational approaches (Alruily, 2012; Althenayan and Menai, 2014; Al-Tuwairesh et al., 2015, 2018; Assiri et al., 2016) while linguistic and stylistic approaches fall short. This calls for a need to contribute to the field of forensic linguistics in general and forensic authorship analysis in Arabic in particular. Social media platforms such as Twitter are heavily populated by users who sometimes abuse such mediums. Saudis are responsible for 30% of the tweets posted (Salim, 2017). Simultaneously, there are efforts to fight cybercrime and issue regulations that incriminate hate speech and offensive language published online.

3 Methodology and corpus design

This section will demonstrate the NASCT corpus design, the data collection process, the sample selected for the study, and the pre-processing of the data and training WEKA.

3.1 Corpus design

Table 1 below shows the breakdown of the Najdi Arabic Specialized Corpus of Tweets.

Author	Tweets	Words
Faisal Alabdulkarim	2,825	60,213
Mansour Alrokibah	2,412	24,727
Abdulrahman Allahim	7,532	135,033
Ali Algofaily	2,424	37,217
Abdullah Alsubayel	5,805	35,791
Abdulaziz Alzamil	4,741	47,832
Taghreed Altassan	5,292	10,560
Wafa Alrasheed	2,200	24,236
Maha Alwabil	1,550	40,263
Arwa Almohanna	1,143	17,956
Ghadah Aleidi	7,952	70,709
Maha Alnuhait	2,089	27,784
Amani Alajlan	12,040	216,027
Total	58,005	748,348

Table 1: NASCT corpus design

3.2 Data collection

The data collected for this study are the authors' posts published on Twitter. To ensure authorship, all original tweets and replies were included in the corpus while retweets were eliminated. The corpus was compiled using Data Miner, a Google Chrome extension that identifies Arabic script. The time period of the data collection was March 1, 2018 – September 30, 2019. The data was produced as Excel sheets which the authors converted into .CSV file format.

3.3 Sample

The sample of the study are six males and seven females. All originated from the central region of Najd and current residents of Riyadh, the capital of Saudi Arabia. In terms of ethical considerations, all authors' accounts are public and verified thereby the tweets they post are public data. Lastly, all authors run the Twitter accounts personally.

3.4 Data preprocessing and training sets

To explore the NASCT using WEKA, the authors had to convert the .CSV files into ARFF file format. In order to train WEKA, the authors reassembled the corpus into thirteen separate ARFF files. They created two data training sets, the first is an unbalanced data set (TS1) which includes the full corpus. As shown in Table1, the subcorpora of some authors are substantially larger in size compared to others. The second one is a balanced data set (TS2), which includes equal number of tweets per author. Both data sets include 80% of the authors' data, and a header describing the types of linguistics attributes being examined. The remaining 20% of the authors' subcorpora was combined into one ARFF file for testing. Table 2 shows the number of tweets per data set.

Unbalanced data set = TS1	Balanced data set = TS2
44192	25247

Table 2: Training data sets

4 Findings and discussion

To examine different classifiers to see which performs most accurately, the authors ran a test using seven classifiers: Linear SVM, Multinominal Naïve Bayes, Decision tree J-48, KNN Depth=3, KNN Depth=5, Random Forest Estimator=5, and Random Forest Estimator=15. Table 3 shows the performance of seven different classifiers in three categories: unigrams, bigrams, and trigrams.

Parameter	Unigram	Bigrams	Trigrams
Linear SVM	0.59	0.6	0.6
M Naïve Bayes	0.47	0.48	0.48
J-48	0.4	0.4	0.4
KNN Depth=3	0.25	0.25	0.25
KNN Depth=5	0.25	0.25	0.25
Random FE=5	0.4	0.42	0.42
Random FE=15	0.46	0.47	0.47

Table 3: N-gram word models per classifier

The results of the first test show that Linear SVM scores the highest accuracy rates, therefore it was implemented in the next stage. The authors ran three tests to explore a range of parameters that can ensure the highest accuracy rates. In the first range, the minimum value is words that appear once and words that appear in 60% in the data files (min_df=1 – max_df=int (60/100)). The second one eliminates words that appear twice or less and words that occur in 80% of the data files (min_df=1 – max_df=int (80/100)). The last parameter test eliminates words that appear once and words that appear in 95% of the data files (min_df=1 – max_df=int (95/100)). Table 4 shows the accuracy rates of different parameters in both data sets.

Parameter	Unigram		Bigrams		Trigrams	
	TS1	TS2	TS1	TS2	TS1	TS2
1–60/100	0.59	0.58	0.6	0.6	0.59	0.6
2–80/100	0.59	0.58	0.6	0.45	0.6	0.29
0.001–95/100	0.49	0.58	0.49	0.59	0.49	0.59

Table 4: Accuracy rates per parameter using Linear SVM

In the first parameter test, both data sets scored the highest accuracy rates. The scores were most accurate across the three n-gram categories (0.59-0.6 respectively). The first data set TS1 scored a consistent and better performance compared to TS2 in the second parameter. The accuracy rates of the balanced data TS2 in the second parameter test were inconsistent. On the other hand, TS2 scored consistently higher results in the third parameter test compared to the unbalanced data set TS1. Nonetheless, both training data sets scored consistent results in unigrams, bigrams, and trigrams.

Furthermore, it appears that the balanced data set scores the highest overall results in unigrams, while the unbalanced data set scores the highest overall results in bigrams. However, the optimum parameter is the first test using bigrams.

5 Conclusion and future studies

This pilot study aimed to explore WEKA in forensic authorship analysis research. To answer the first question, the authors found that the unbalanced data set performed better than the smaller, balanced one. The large the size of the data provides WEKA with training and recognizing the features more accurately. As for which classifier performs best, results show that Linear SVM has the most accurate performance. This conforms to the findings of Fissette (2010) and Braocardo et al. (2014). Lastly, the results show that bigrams can accurately identify authorship, which confirms the findings of Feiguina and Hirst (2007).

For future studies, implementing different proportions for training and testing might yield higher, more accurate rates.

References

Alothman, E. 2012. *Digital Vernaculars: An Investigation of Najdi Arabic in Multilingual Synchronous Computer-Mediated Communication*. PhD thesis. University of Manchester.

Alruily, M. 2012: Saudi tweets dataset. figshare. Dataset.

Alshutayri, A and Atwell, E. 2018. Creating an Arabic Dialect Text Corpus by Exploring Twitter, Facebook, and Online Newspapers. In *Proceedings of OSACT'2018 Open-Source Arabic Corpora and Processing Tools. OSACT'2018 Open-Source Arabic Corpora and Processing Tools*, pages 07-12 May 2018, Miyazaki, Japan. (In Press)

Althenayan A. S. and Menai M.E-B. 2014. Naïve Bayes classifiers for authorship attribution of Arabic texts, *Journal of King Saud University - Computer and Information Sciences*, 26, pages 473-484.

Banga, R., Bhardwaj, A., Peng, S. L., & Shrivastava, G. 2018. Authorship Attribution for Online Social Media. In *Social Network Analytics for Contemporary Business Organizations*, pages 141-165. IGI Global.

Binturki, T. 2015. *The Acquisition of Negation in Najdi Arabic*. PhD thesis. University of Kansas.

Bloch, B. 1948. A set of postulates for phonemic analysis. *Language,* 24, pages 3-46.

Brocardo, M. L., Traore, I., and Woungang, I. 2014. Toward a framework for continuous authentication using stylometry. In *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, pages 106-115. IEEE.

Cotterill, J. 2010. How to use corpus linguistics in forensic linguistics. In O'Keeffe, A and McCarthy, M., eds., *The Routledge Handbook of Corpus Linguistics*. London: Routledge, pages 66-79.

Crystal, D. 1997. *The Cambridge Encyclopedia of Language*. Cambridge: Cambridge University Press.

Feiguina, O. and Hirst, G. 2007. Authorship attribution for small texts: literary and forensic experiments. Paper presented to *the International Workshop on Plagiarism Analysis, Authorship Identification and Near-Duplicate Detection*. 30th Annual International ACM SIGIR (SIGIR '07).

Fissette, M. 2010. Author identification in short texts.

Juola, P. 2008. Author Attribution, Foundations and Trends in Information Retrieval. (In Press)

Koester, A. 2010. Building small specialised corpora. In O'Keeffe, A and McCarthy, M., eds, *The Routledge Handbook of Corpus Linguistics*. London: Routledge, pages 66-79.

Koppel, M., Schler, J. and Argamon, S. 2009. "Comp utational Methods in Authorship Attribution.". *Journal of the American Society for Information Science and Technology*, 60(no. 1), pages 9–26.

Rico-Sulayes, A. 2011. Statistical authorship attribution of Mexican drug trafficking online forum posts. *International Journal of Speech, Language & the Law*, *18*(1).

Saha, N., Das, P., & Saha, H. N. 2018. Authorship attribution of short texts using multi-layer perceptron. *International Journal of Applied Pattern Recognition*, *5*(3), pages 251-259.

Salim, F. 2017. *The Arab Social Media Report 2017: Social Media and the Internet of Things: Towards Data-Driven Policymaking in the Arab World* Dubai: MBR School of Government. Vol. 7.

Shrestha, P., Sierra, S., González, F. A., Montes, M., Rosso, P., & Solorio, T. 2017. Convolutional neural networks for authorship attribution of short texts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 669-674.

Banga, R., Bhardwaj, A., Peng, S. L., & Shrivastava, G. 2018. Authorship Attribution for Online Social Media. In *Social Network Analytics for Contemporary Business Organizations*, pages 141-165. IGI Global.

Turell, M. T. 2010. The use of textual, grammatical, and sociolinguistic evidence in forensic text comparison. *The International Journal of Speech, Language and the Law*, 17(2), pages 211-250.

Twitter. About Twitter Verified Accounts. *URL: https://help.twitter.com/en/managing-your-account/about-twitter-verified-accounts*.

WEKA.Witten, I. H., Frank, E., and Hall, M. 2016. *Data Mining: Practical Machine Learning Tools and Techniques*. San Francisco: Morgan Kaufmann Publishers, fourth edition.

Native-Language Identification with Attention

Stian Steinbakken and **Björn Gambäck**
Department of Computer Science
Norwegian University of Science and Technology
Trondheim, Norway
stiansteinbakken94@gmail.com, gamback@ntnu.no

Abstract

The paper explores how an attention-based approach can increase performance on the task of native-language identification (NLI), i.e., to identify an author's first language given information expressed in a second language. Previously, Support Vector Machines have consistently outperformed deep learning-based methods on the TOEFL11 data set, the *de facto* standard for evaluating NLI systems.

The attention-based system BERT (Bidirectional Encoder Representations from Transformers) was first tested in isolation on the TOEFL11 data set, then used in a meta-classifier stack in combination with traditional techniques to produce an accuracy of 0.853. However, more labelled NLI data is now available, so BERT was also trained on the much larger Reddit-L2 data set, containing 50 times as many examples as previously used for English NLI, giving an accuracy of 0.902 on the Reddit-L2 in-domain test scenario, improving the state-of-the-art by 21.2 percentage points.

1 Introduction

Native-language identification (NLI) is the task of identifying an author's first language (L1; e.g., Spanish) given only information expressed in the author's second language (L2; e.g., English). NLI operates under the assumption that an author's L1 will dispose them towards particular language production patterns in their L2 (MacDonald, 2013). Knowing what mistakes a learner typically makes when writing or speaking in a foreign language can help educators create custom tutoring experiences and give feedback based on L1s. NLI also has applications within forensic linguistics, where identifying what country an unknown author is originally from is useful when detecting, e.g., plagiarism, web fraud and grooming, as well as in web-applications, web-crawling, and data collection to ensure high quality data.

Containing English learners of 11 different L1s, the TOEFL11 data set (Blanchard et al., 2013) has been the standard corpus for NLI. However, recent advances in auto-generating data based on social media users of high proficiency (Goldin et al., 2018) have enabled the collection of much larger data sets with more languages. State-of-the-art approaches on TOEFL11 reach over 0.88 accuracy in identifying the native language of the author using text only (Cimino and Dell'Orletta, 2017), while the best results on more English-proficient social media users approach 0.69 when evaluated on the same topics as trained on (Goldin et al., 2018). However, today's state-of-the-art systems are known for quite substantial drops in performance when tested on documents about topics not seen during training (Malmasi and Dras, 2018).

While the NLI field is evolving, so are advances in deep learning. Recently, attention-based models have shown promising results on various NLP tasks, and have become the *de facto* standard in sequence-to-sequence processing. Models such as the Transformer (Vaswani et al., 2017) rely solely on attention mechanisms, allowing for heavy parallelisation in the model training, as no recurrence or convolution is required. Using a network of transformers, BERT (Bidirectional Encoder Representations from Transformers) obtained new state-of-the-art results on 11 natural language processing tasks varying from question answering to sentence classification (Devlin et al., 2018).

Given the success of attention-based systems and the need for good NLI systems, the paper explores how such an approach can improve NLI performance, and investigates how robust attention-based systems are when tested on different topics than those they were trained on. Furthermore, while such systems perform well on their own, the paper also addresses how they can allow for improvements in combination with existing tech-

Proceedings of the 17th International Conference on Natural Language Processing, pages 261–271
Patna, India, December 18 - 21, 2020.

niques, given that all the best NLI systems to date include some ensemble or multi-classifier based architecture.

The paper is structured as follows: Section 2 describes the main data sets used in the field of NLI, while Section 3 covers related work. Section 4 introduces the model architectures, before Section 5 provides an overview of the experiments and their results. Finally, Section 6 sums up the findings and provides suggestions for further study.

2 Data Sets

The field of Native Language Identification is in a broader perspective quite young, and only gained serious momentum during the previous decade, most notably after Koppel et al. (2005) trained a Support Vector Machine (SVM) model on a vast amount of features including part-of-speech (POS) tags, n-grams and grammatical errors on ICLE, the International Corpus of Learner's English (Brooke and Hirst, 2013). Since then, two shared tasks have been dedicated to NLI, in 2013 and 2017. The experiments below and the discussed related work all focus on two data sets, TOEFL11 (the main data set used in the shared tasks) and Reddit-L2:

The **TOEFL11 data set**[1] (Blanchard et al., 2013) consists of English essays written by people with 11 different first languages for the college-entrance Test of English as a Foreign Language: Arabic, Chinese, French, German, Hindi, Italian, Japanese, Korean, Spanish, Telugu and Turkish. The corpus contains 13,100 essays, with 1,100 essays per language distributed as evenly as possible across 8 different topics. Each essay is also labelled with its score (low, medium or high). For the 2013 shared task, the data set was split into a training set consisting of 900 essays per L1, and development and test sets consisting of 100 essays each per L1.

The **Reddit-L2 data set**[2] was collected by Rabinovich et al. (2018) from the social media platform Reddit, where users can subscribe to areas of interest, called "subreddits" (Singer et al., 2014). These subreddits vary from general topics such as "pictures" to specific niche areas such as "birds with arms" (which is for posting pictures of birds photoshopped to look like they have arms). Some subreddits are dedicated to Europe, where the users can optionally provide a tag ('flair') indicating their home country. These flairs were utilised by Rabinovich et al. to infer the users' L1, with experiments performed to quantify the reliability of the labels. Goldin et al. (2018) added further measures to verify the labels' correctness.

While TOEFL11 contains texts written by learners of English, many of the Reddit-L2 authors are close to fluent in English, making the task of performing NLI on this data set more challenging. The data set consists of texts the users have produced in Europe-related subreddits (*in-domain*), as well as texts written in other subreddits, that can have virtually any topic (*out-of-domain*). The data is separated into chunks of 100 sentences from one user. After pre-processing and removing users with less than 100 sentences, the Reddit-L2 data set consists of roughly 200 million sentences and 3 billion tokens across 29 native countries, from 34,511 unique users.

The number of users varies between countries, making the data set imbalanced. Hence Goldin et al. (2018) downsampled the data, grouping 29 native countries into 23 languages: English, German, Dutch, French, Polish, Romanian, Finish, Swedish, Spanish, Greek, Portuguese, Estonian, Czech, Italian, Russian, Turkish, Bulgarian, Croatian, Norwegian, Hungarian, Lithuanian, Slovenian and Serbian. The number of users per label was capped at 104, which is the number of users present for the labels with fewest users (Lithuania and Slovenia). For labels with more than 104 users available, 104 users were selected at random. For each user, the median number of chunks was selected. 3 chunks per user for an in-domain scenario, and 17 chunks per user for an out-of-domain scenario,

3 Related Work

As mentioned in the previous section, two shared tasks have been dedicated to NLI, taking place in 2013 and 2017, and with respectively 29 and 19 teams participating. The **2013 NLI Shared Task** (Tetreault et al., 2013) introduced the TOEFL11 corpus and was divided into three sub-tasks: Closed-Training, Open-Training-1 and Open-Training-2. The first was the main task, allowing usage of the training and development sets only. The Open-Training-1 task allowed the use of any training data *except* TOEFL11, while Open-Training-2 allowed the use of any data, including TOEFL11. However, Open-Training-1 showed that training on external corpora while testing on TOEFL11 caused a significant drop in accuracy.

[1] catalog.ldc.upenn.edu/LDC2014T06
[2] cl.haifa.ac.il/reports/L2/index.shtml

The most common features used were word, character and part-of-speech n-grams. Four of the top five teams used at least word 4-grams, and some as high as 7- and 9-grams. The best team achieved an accuracy of 0.846 (Tetreault et al., 2013), and like the overwhelming majority of the participating teams used SVMs. Tetreault et al. (2012) additionally improved performance using ensemble methods. Later, Ionescu et al. (2014) improved the results further using string kernels.

The **2017 NLI shared task** (Malmasi et al., 2017) was similar to the 2013 task, but in addition to the text utilised TOEFL11 data from a speech-based NLI task included in the 2016 INTERSPEECH Computational Paralinguistics Challenge (Schuller et al., 2016). The raw speech data could not be distributed in that challenge, so the data set consisted of textual transcripts together with so called i-vectors, which are vectors of fixed length (here 800), working as lower-dimensional representations of high-dimensional sequential speech recordings.

Hence the 2017 shared task was divided into three tracks: essays only, speech only, and a combined 'fusion' track using both text and speech. The fusion track submissions showed that combining written and spoken responses provided a large boost in prediction accuracy (Malmasi et al., 2017). Furthermore, ensemble-based systems were the most effective in all tasks, but typically the same features were used as in 2013, and SVMs were still the most popular approach, dominating deep learning models. Ircing et al. (2017) hypothesise that this could be due to the size of the TOEFL11 data set, and that more training examples could help deep learning models perform better.

UnibicKernel (Ionescu and Popescu, 2017) performed best on the fusion track (0.932 accuracy),[3] topped the speech-only track, and placed in the top tier in the essay track, by using multiple kernel learning and kernel discriminant analysis, KDA. KDA is a kernelised version of LDA, Linear Discriminant Analysis, that is, a method for finding the best linear combination of features that characterise or separate two or more classes, by projecting the data points down from a feature space where they not are linearly separable to a lower dimensional space where they are.

[3]The 2017 shared task used macro-averaged F_1 score as the official evaluation metric, but also reported accuracy. However, accuracy will be used here for consistency, and since the F_1 scores were typically almost identical to the accuracy.

Also finishing in the top group was CEMI (Ircing et al., 2017), using a neural network based meta-classifier of several isolated feed-forward neural network (FFNN) models, each trained on a separate feature type (such as word, character and POS n-grams, plus i-vectors), with the outputs combined using softmax to predict the final label.

The ItaliaNLP team (Cimino and Dell'Orletta, 2017) topped the essay-only track with 0.882 accuracy, which currently is the highest score achieved on the textual TOEFL11 test set. They used two stacked SVM-classifiers, one trained at sentence level and the other at document level, using the output of the sentence classifier as input, together with features from the documents themselves: text and average word length; function words; as well as character, word, lemma, POS, and linear dependency n-grams.

After the shared tasks, Malmasi and Dras (2018) performed a systematic examination of ensemble methods for NLI, in addition to evaluating deeper ensemble architectures such as classifier stacks. The experiments included a rigorous application of meta-classification models, achieving state-of-the-art results on several large data sets, evaluated both intra-corpus and cross-corpus. Two important trends were observed: the meta-classification results were better than the ensemble combination methods alone, and meta-classifiers trained on continuous output performed better than their discrete label counterparts. The best performing meta-classifier was LDA, both individually and as an ensemble. The ensemble of LDA meta-classifiers obtained an accuracy of 0.871 — close to the state-of-the-art accuracy of 0.882 obtained by ItaliaNLP.

Goldin et al. (2018) tried the NLI task in a new environment using the larger **Reddit-L2** data set. A simple regression classifier was trained for all experiments, since the main focus was not to build a new NLI classifier, but to explore possible features specific to social media and the Reddit-L2 data. They used content features (3-grams and POS 1-grams), spelling and grammar features, and content-independent features such as function words, the most frequent POS 3-grams in the data set, and average sentence length. Furthermore, Goldin et al. experimented with social network specific features such as votes from other users, average number of submissions and comments, and the most frequent subreddits the user had visited, compared to the most popular subreddits for each country.

The most visited subreddits feature performed stunningly, both in and out of the domain. However, this feature is not only specific to the particular data set, but as many users tended to frequent subreddits specific to their country, the feature often contained the correct label itself. Hence Goldin et al. reported the results of using all features, excluding the Subreddit feature, which yielded an accuracy of 0.690 in the in-domain scenario, dropping down to 0.36 when tested out-of-domain.

4 Architecture and Model

Three model architectures were used in the experiments below: a stand-alone BERT model tailored to NLI, a meta-classifier architecture, and an ensemble of meta-classifiers. The stacked classification architectures use traditional techniques in combination with BERT. The PyTorch version of BERT was used,[4] which is a clone of Google's official BERT implementation for TensorFlow,[5] using the same pre-trained models as provided by Devlin et al. (2018). All other classifiers were implemented using the open source Scikit-Learn (Sklearn) Machine Learning library.[6] For SVMs the default parameters of Sklearn were used, but with a linear kernel, as it was the kernel used for all base classifiers in Malmasi and Dras (2018). The FFNNs (feed-forward neural networks) were run with the default parameters of Sklearn, unless specified otherwise. Sklearn was also used for feature manipulation (e.g., transforming text into TF-IDF weighted vectors), with Natural Language Toolkit (NLTK)[7] used for basic text-processing, and Pandas[8] for data manipulation.

All experiments were carried out on a two GPU cluster, with 16 GB each. For most jobs, 64 GB RAM was sufficient, but for the largest job 128 GB RAM was needed in order to keep both BERT and all the data in memory. Running BERT on TOEFL11 for 5 epochs typically took 2–3 hours, while experiments using millions of Reddit-L2 examples took more than 6 days. All the code from the paper is available on GitHub.[9]

[4] github.com/huggingface/pytorch-pretrained-BERT
[5] github.com/google-research/bert
[6] scikit-learn.org/stable/
[7] www.nltk.org/
[8] pandas.pydata.org/
[9] github.com/stianste/BERT-NLI

4.1 BERT Model Architecture

The overall BERT architecture consists of the pre-trained BERT model followed by a linear layer, as suggested by Devlin et al. (2018). The linear layer is randomly initialised and must be trained from scratch for each classification task. It has an input size the same as the BERT hidden size output and an output size matching the number of labels for the relevant task (11 for TOEFL11 and 23 for Reddit-L2). The BERT model and the linear layer are trained together using the cross-entropy loss with regards to the correct label.

To reduce vocabulary size and tokenisation complexity, the uncased version of BERT was used in all experiments, although including capitalisation could in some cases be beneficial for NLI. Following Devlin et al. (2018), all experiments utilised the same hyper-parameter settings, except for three that were varied in the ranges found by them: batch size $\{16, 32\}$, Adam learning rate $\{5e^{-5}, 3e^{-5}, 2e^{-5}\}$, and number of epochs $\{3, 4\}$, with fixed numbers of hidden layers (12), hidden size (768), attention heads per transformer (12), proportion of training to perform linear learning rate warmup for (0.1), and gradient accumulation steps (1).

However, while Devlin et al. restricted the maximum total input sequence length after WordPiece tokenisation to 128, it was here set to its largest possible value, 512, since most documents available in the NLI data sets contain far more tokens than 128: the average and maximum number of WordPiece tokens per document is 369 and 910 for TOEFL11, growing to 2,072 and 18,149 for Reddit-L2. 98.9% of the TOEFL11 documents contain more than 128 tokens, but only 4.9% have more than 512 tokens.

In the Reddit-L2 data set, however, all documents surpass 512 tokens, so it must be split up into chunks, in order not to waste data. A heuristic division of the documents was applied, dividing all examples into smaller sub-examples (sub-chunks) with a length of roughly 512 tokens, by splitting on spaces. After training has been performed on the sub-chunks, evaluation was carried out both on the individual sub-chunks and on the recombination back into the original chunks, using a majority vote based on each sub-chunk prediction. Ties were broken randomly.

It is important to note that the previous work on Reddit-L2 by Goldin et al. (2018) evaluated on the prediction of each chunk. For this reason, the accuracies obtained at the chunk level will be the

comparable measurement of the system's performance, not the *ad hoc* sub-chunks that have only been created in order to fit the limitations of BERT.

Preliminary experiments further showed that BERT-large with a maximum sequence length of 512 caused memory issues with the GPUs available, as did a batch size of 32 for BERT-base. The largest models that were able to run with a 512 sequence length were BERT-base with a batch size of 16 or less, and BERT-large with a batch size of 1. Hence all experiments used the BERT-base model, except when running BERT-large with batch size 1. This is unfortunate, as Devlin et al. (2018) report that BERT-large provides a significant performance boost over BERT-base, even for small data sets.

4.2 Meta-Classifier Architectures

To explore how an attention-based system can be used in union with the current-state-of-the art approaches to improve NLI performance, two novel architectures were tested, inspired by the classifier-stacking of Malmasi and Dras (2018), but including BERT as an optional base-classifier.

The first is a stacked architecture consisting of homogeneous base classifiers, with their combined output fed into a meta-classifier providing the final decision. An illustration of the meta-classifier architecture can be found in Figure 1. The inclusion of BERT as a base-classifier is optional, indicated by stippled lines. All base classifiers are fed the raw text input, but trained on different feature types.

After training, each base classifier produces a vector of continuous probabilities for each class. This is used since Malmasi and Dras (2018) showed continuous probability output in general yielding better performance than discrete one-hot encoded label representation. The meta-classifier is then trained on the concatenated output of the base classifiers and the original training labels, to produce a single prediction per example instance.

The second architecture is similar to the first, but instead of using a single meta-classifier, an ensemble of meta-classifiers is applied to provide the final decision of the stack. It follows the best performing ensemble found by Malmasi and Dras (2018), with 200 meta-classifiers merged in a bagging-ensemble, where all models are trained on different subsets of the base classifiers.

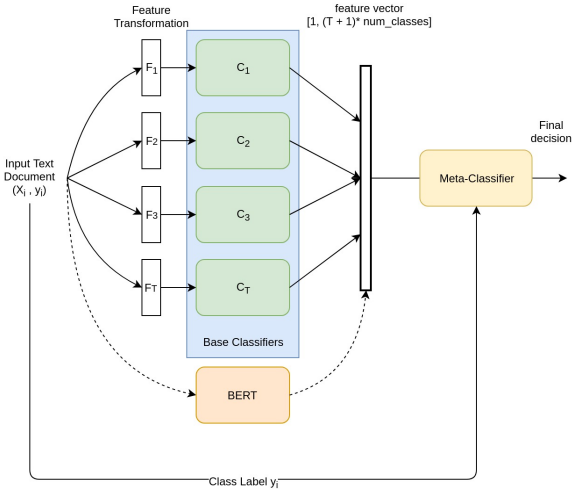


Figure 1: A meta-classifier stack with BERT as an optional base classifier

5 Experiments and Results

To set reasonable baselines for the experiments, a simple multinomial Naïve Bayes (MNB) and an SVM classifier were trained and evaluated on the unigram representation of the documents. As done by Malmasi and Dras (2018), the unigram feature vector was normalised using each word's TF-IDF score. Initial experiments were then run to give an indication of how an attention-based architecture in isolation performs on the NLI task, as well as when tested on documents on topics different from what it was initially trained on. Further experiments addressed whether the combination of an attention-based system and the techniques used in the current state-of-the-art can improve performance. Finally, it was investigated how the attention-system performs with more data available.

5.1 Attention-based System Only

The first set of experiments aimed to explore how an attention-based system alone performs on the NLI task. These initial experiments also indicated what hyper-parameters the BERT model performs best under. As most of the related literature trains on the TOEFL11 training set and evaluates on the official TOEFL11 test set, that setup was applied. Similarly, experiments on Reddit-L2 in-domain data set applied the same downsampling and 10-fold cross-validation as used by Goldin et al. (2018). To test how an attention-based system performs on the task of NLI when tested on documents concerning topics different from what it was initially trained on, the out-of-domain experimental setup

MNB	SVM	BERT			
		$2e^{-5}$	$3e^{-5}$	$4e^{-5}$	$5e^{-5}$
0.559	0.726	0.759	**0.777**	0.761	0.765

Table 1: TOEFL11 accuracies, different learning rates

described in Goldin et al. (2018) was followed, i.e., training the model on the in-domain data, and testing on out-of-domain data of different users.

The results of the experiments on **TOEFL11** are found in Table 1. As expected, the SVM classifier clearly outperforms Naïve Bayes. The best results using BERT were obtained with a learning rate of $3e^{-5}$, with a final accuracy of **0.777**. Under these hyper-parameters, the model obtained a training loss of 0.110 and an evaluation loss of 0.824. Experiments were run over 3, 4, 5 and 10 epochs, but the table only reports the results after 5 epochs, since increasing epochs typically increased accuracy, regardless of the learning rate. However, the 10-epoch model achieved an accuracy slightly below the best at 5 epochs, with a training loss of 0.004, but an evaluation loss of 1.140 on the test set, indicating that it overfitted the training data.

The BERT experiments reported in Table 1 were run with a constant batch size of 16. Testing different batch sizes $\{1, 2, 3, 4, 8\}$ under the optimal settings (a learning rate of $3e^{-5}$ over 5 epochs) gave no indication that a higher or lower batch size is better or worse, so the remaining experiments were run with size 16 batches, in order to keep the training time as low as possible without encountering memory issues or reducing model performance.

For comparison, an experiment running BERT-large with a batch size of 1 was also carried out, using the best performing learning-rate and number of epochs found for BERT-base. However, BERT-large was not able to converge under these settings: after 5 epochs it produced the same probability for each L1 for all test cases. Hence BERT-large was run with both a smaller and larger learning rate. While larger learning rates did not improve matters, a learning-rate of $2e^{-5}$ produced a final accuracy on the TOEFL11 test set of 0.759. BERT-base with a batch size of 1 obtained an accuracy of 0.770, so BERT-large does not seem to provide any significant benefit over BERT-base for TOEFL11.

Table 2 shows average accuracy over 10-fold cross-validation on the **Reddit-L2** data set. After downsampling the data set and splitting all documents into sub-documents of max 512 tokens,

Scenario	MNB	SVM	BERT
In-domain, chunks	0.377	0.716	**0.805**
In-domain, sub-chunks	0.350	0.574	0.651
Out-of-domain, chunks	0.176	0.400	**0.502**
Out-of-domain, sub-chunks	0.169	0.322	0.400

Table 2: Reddit-L2 accuracies

there were roughly 18,300 training examples and 1,943 test cases per fold. Both sub-chunk and chunk/document accuracy are reported. After recombining all sub-chunks, there was an average of 500 test chunks per fold.

As expected based on related work, the simple unigram SVM baseline improves on the 0.690 accuracy Goldin et al. (2018) obtained on the in-domain scenario using logistic regression. The unigram SVM with an accuracy of 0.400 also beats the previous best out-of-domain score of 0.362. The accuracy of both Naïve Bayes and SVM drop when tested in the out-of-domain Reddit-L2 scenario, and further drop when the models are evaluated on the more granular sub-chunks. The final average in-domain accuracy of BERT across the ten folds was **0.805**, a substantial improvement over the SVM classifier and over the current state-of-the-art (0.690). When evaluating on each individual sub-chunk, the accuracy drops to 0.651, indicating that the model predicts parts of the chunks incorrectly, but performs well on the documents/chunks as a whole using majority vote.

Running BERT in the out-of-domain Reddit-L2 scenario, the final average over random 10-fold cross-validation was **0.502**, a clear improvement over the single SVM baseline of 0.400, and again a substantial improvement over the 0.362 obtained by Goldin et al. (2018). Note that the train-test split in the out-of-domain scenario can be regarded as unfavourable: After downsampling, the 10% out-of-domain test chunks outnumber the 90% in-domain training chunks, with most folds having 16,000–16,500 training examples, but 20,000–21,000 test cases. Commonly, training sets are 5–10 times as big as test sets, but with the out-of-domain scenario the training set is smaller than the test set, making for a rigorous evaluation scenario.

Comparing the out-of-domain results to the in-domain results, BERT's accuracy drops from 0.805 to 0.502, a 37.9% relative drop in accuracy, compared to the 47.5% drop obtained by Goldin et al. (2018), indicating that BERT is more robust than logistic regression when tested off-topic.

Base Classifier / Meta-Classifier	SVM	FFNN
SVM	0.756	0.745
SVM + BERT	0.794	0.791
FFNN	0.819	0.825
FFNN + BERT	0.838	**0.853**

Table 3: Meta-classifier results on TOEFL11

Base Classifier / Ensemble	SVM	FFNN
SVM	0.755	0.801
SVM + BERT	0.798	0.823
FFNN	0.827	0.808
FFNN + BERT	0.849	**0.851**

Table 4: Ensemble of meta-classifiers on TOEFL11

5.2 Combining Attention with State-of-the-Art Techniques

The next set of experiments utilised the meta-classifier and ensemble architectures described in Section 4.2. A grid search over several hyper-parameters was carried out, over the different base classifiers, the types of features per classifier, and the maximum number of features per classifier. The different models and feature types were first explored individually, in order to assess their individual contribution to the meta-classifier or ensemble. Both SVMs and FFNNs were used as base classifiers. Based on the features used by Ircing et al. (2017) and Malmasi and Dras (2018), word 1–3 grams, character 1–4 grams, and lemma 1–2 grams were used, in addition to content-independent function word 1–2 grams. Similarly to Malmasi and Dras (2018), the main focus was on evaluating whether the inclusion of BERT can improve the results obtained by using a predefined set of features, rather than on feature exploration.

The SVM and FFNN base classifiers were run with the TF-IDF weighted representation of each feature-type, with the maximum number of features per example capped at 5,000, 10,000, 30,000, and no limit. Word 3-grams performed surprisingly bad compared to word 1- and 2-grams, and the content-independent function word features performed far worse than the content-dependent features. The single best performing feature type using both SVM and FFNN was character 4-grams. The SVM model favoured feature vectors of size 30,000, while the FFNN performed better with no feature limit, potentially since the FFNN is better at filtering out the noise obtained by including more TF-IDF features, while still finding useful information in more than 30,000 features.

The results of training a **meta-classifier** on the continuous probability outputs of the ten base classifiers from the previous experiment are found in Table 3. The inclusion of BERT means appending the $n\_classes$ logit outputs of BERT to the training and test data. Experiments were also carried out normalising BERT's output using softmax and raw probabilities, but using the raw logit output performed better overall.

The inclusion of BERT has a significant impact on the performance: using SVMs as base and meta-classifier, the accuracy increases from 0.756 to 0.794. The FFNN trained on the SVM base classifiers reap even more benefits when including BERT, increasing from 0.745 to 0.791. Perhaps surprisingly, FFNNs perform best both as base-classifier and meta-classifier: the accuracy obtained by using FFNN base classifiers and BERT with a FFNN meta-classifier is **0.853** (up from 0.825), slightly below the current best text-only score on TOEFL11 test, 0.882 by Cimino and Dell'Orletta (2017).

The results of training an **ensemble of meta-classifiers** can be found in Table 4. Again, the inclusion of BERT causes a significant 4.3 percentage point increase for the best performing ensemble. However, contrary to the results of Malmasi and Dras (2018), there was no clear gain in performance when applying a single meta-classifier as opposed to using a bagging ensemble of meta-classifiers. This is surprising and interesting, as the experiments and setup are quite similar.

Looking more closely at the results of Malmasi and Dras, though, the increase in accuracy when going from a single meta-classifier to an ensemble of meta-classifiers was only present for the LDA meta-classifier, while the other meta-ensembles (e.g., the SVM-based one) obtained the exact same TOEFL11 test set accuracy as their corresponding meta-classifiers. Thus, the lack of increase in performance when using an ensemble of meta-classifiers as opposed to using a single one could actually be expected for the SVM and FFNNs used.

Experiments using LDA as the meta-classifier were also carried out, but provided disappointing results, a lot lower than the best single base classifier, potentially due to the collinearity of the features used. Using a more diverse feature set might have made LDA a more viable candidate, both as a single meta-classifier and in an ensemble.

Classifier / Ensemble	In-domain	Out-of-domain
FFNN meta-classifier	0.765	0.452
FFNN + BERT	**0.818**	**0.529**
BERT alone	0.805	0.502

Table 5: Ensemble of meta-classifiers on Reddit-L2

The stack setup which performed best on TOEFL11 (i.e., the FFNN meta-classifier trained on the outputs of the base FFNN classifiers and BERT) was run also on the **Reddit-L2** in- and out-of-domain scenarios (Table 5). Using the same 10 folds as the previous in-domain experiment, the ensemble *without* BERT obtained an accuracy of 0.765. The same ensemble *with* BERT boosted the accuracy to **0.818**, Hence BERT seems to be doing most of the heavy lifting in the ensemble in the in-domain scenario. However, the ensemble also provides BERT with a minor performance boost, increasing accuracy by 1.3 percentage points compared to using BERT alone on the task.

The final average accuracy over the 10 out-of-domain folds obtained by the meta-classifier was 0.452 without using BERT. Including BERT in the ensemble yielded an average accuracy of **0.529**. This seems comparable to the in-domain results, as the final meta-classifier accuracy again is slightly higher than that obtained by using BERT alone.

5.3 More Data

While the Reddit-L2 data set is huge compared to previous NLI data sets, most of the data is discarded when performing downsampling, which is done to maintain class balance. Using evaluation metrics that take class imbalance into consideration, such as the macro-averaged F_1 score, can mitigate this problem. Hence BERT was trained on the out-of-domain Reddit-L2 data, and then the entire in-domain data set was used for testing, to show how the same attention-system as used alone in the first experiments performs with more data available. Training on only out-of-domain data will also display how the system reacts when trained on some topics and tested on others.

To maintain a fair training-to-test ratio, the out-of-domain training data was engineered to be roughly 10 times the size of the in-domain test set, and was balanced to contain roughly the same number of examples per label. This was done by capping the maximum number of training sub-chunks per label to 80,000, leaving 1,491,198 sub-chunks for training, with 282,385 for testing. This heuristic division causes some class imbalance in the training data, as some languages have less than 80,000 sub-chunks, with Slovenian only having 24,787 sub-chunks available out-of-domain. This issue will have to be tolerated, however, as the intent of the experiment was to explore how more data impacts performance. Using macro-averaged F_1 as evaluation metric will also account for this bias to some extent, by measuring performance with the same weighting for each class, An alternative solution would be to set the cap at 24,787 sub-chunks per class, leaving only 570,101 training instances.

After recombining the sub-chunks into the original chunks, there were a total of 71,716 test chunks, far more than the roughly 500 test chunks per fold in the in-domain scenario of Goldin et al. (2018) and the previous experiments. The final accuracy obtained by BERT on these test chunks in the in-domain Reddit-L2 data set was **0.861**. For comparison, the Naïve-Bayes unigram baseline classifier achieved an accuracy of **0.278** when trained and tested on the same data, while the SVM classifier failed to finishing training. As the in-domain test set is not fully balanced with regards to classes, with English having 27.8% of the test labels, the accuracy achieved might have been artificially high. However, the final macro-average F_1 score obtained was **0.847**, indicating that the model is performing well over all classes.

Inspecting the confusion matrix of the predicted outputs, the model seemed to be slightly biased towards predicting English, in particular when the correct label was French or Dutch. Interestingly, the model confused Norwegian with Swedish, and vice versa, both of which were also mistaken for English in 4% of the cases. As for the classes with the fewest training and test instances, such as Serbian, Slovenian and Lithuanian, the model seemed to have no problem, as it obtained 0.86, 0.85 and 0.91 accuracy internally within those classes. In fact, both Turkish and Estonian have rather few training examples, and less than 1,600 test cases, but the model achieved 0.98 and 0.94 in classifying these labels, respectively.

When evaluating this model on the same 10 folds as used in the in-domain scenario in the first experiment, it obtained a final accuracy of **0.902**, and an F_1 score of **0.901**. This task can be considered easier than the task of predicting the in-domain data set as a whole, as the average number of chunks per fold were roughly 500, as opposed to the 70K

test cases when testing on the entire in-domain set. However, the 10 downsampled folds are balanced for classes, making the task non-trivial. The accuracy of 0.902 indicates that BERT thrives with more data available, even though the out-of-domain examples used for training contain no specific topic.

6 Conclusion

The paper is the first to apply BERT to the native language identification task. An empirical exploration showed that BERT alone is not able to compete with traditional state-of-the-art approaches on the TOEFL11 data. However, a meta-classifier architecture combining BERT and ten traditional classifiers trained on different features produced an accuracy of **0.853** on the TOEFL11 test set, closing in on the current state-of-the-art of 0.882.

Further experiments showed that BERT can produce state-of-the-art results on the novel Reddit-L2 data set, both in- and out-of-domain, with a **0.902** 10-fold cross-validated accuracy for the in-domain scenario, with a model trained only on out-of-domain data, but tested on in-domain data. This can be compared to the result of 0.690, obtained by Goldin et al. (2018) when both training and testing the model on in-domain data. With the same setup, BERT achieved an accuracy of 0.805.

Additionally, when evaluated on the entire in-domain test set as a whole (70,000 test cases versus roughly 500 test cases for each fold in the previous in-domain scenario), BERT was able to obtain an even higher accuracy of 0.861. This is not directly comparable to any related work, as the entire in-domain part of the data set has not before been used for testing; however, this task is potentially more demanding than the original in-domain task, as it contains 140 times as many test cases in each fold.

Testing the BERT model out-of-domain showed that it is more robust to topic differences than previous approaches on Reddit-L2, both BERT alone and in a meta-classifier stack, which produced a final accuracy of **0.529**, a 16.7 point improvement over the state-of-the-art.

The performance of BERT with more data available indicates that BERT's performance increases with more data, regardless of what topics it is trained on. In addition to the document granularity the model is trained on, a further reason for why BERT performs so much better on Reddit-L2 than on TOEFL11 can be the number of spelling mistakes found in the TOEFL11 data. This is due both to the higher proficiency of the Reddit-L2 users and to them having spellcheckers and other tools available. A higher frequency of misspelled words can cause the WordPiece representations used by BERT to be different from those which it was pre-trained on. This might be mitigated by further training of the pre-trained models from BERT's check points on data containing spelling mistakes, providing a custom BERT model with embeddings tuned to include spelling mistakes.

As BERT obtained such promising results on the Reddit-L2 in-domain data set, future work should look into how to increase the accuracy on TOEFL11, the standard data set for NLI. In addition of further pre-training of BERT to learn embeddings of spelling errors and other domain attributes, this would include using additional information sources, such as the sentiment and emotions reflected in the texts, as done by Markov et al. (2018b), or including information about punctuation (Markov et al., 2018a) or capitalisation. Clearly, training on more relevant data would also improve performance, so including training data from the italki corpus (Hudson and Jaf, 2018) should be feasible. It contains about 122,000 documents gathered from the language learning site italki, in the same languages as TOEFL11.

With more power available, running BERT-large with a sequence length of 512 should be feasible, or alternatively extensions of BERT or similar models, such as ALBERT (A lite BERT; Lan et al., 2020), GPT-3 (Generative Pre-trained Transformer; Brown et al., 2020) continuous pre-training ('ERNIE 2.0'; Sun et al., 2020) or transformers for longer sequences ('BigBird'; Zaheer et al., 2020) could be tested on the problem.

Opening up the attention mechanism and looking at what parts of the input sequence the model is paying attention to could also give new insights, and could potentially help educators.

Acknowledgments

Thanks to Hans Olav Slotte, Iselin Eriksen, Charles Edvardsen and Vebjørn Isaksen for good discussions on NLI and BERT.

Further thanks to Rabinovich et al. for making the Reddit-L2 data set openly available online, to Devlin et al. for distributing the pre-trained BERT models and their code open source, and to Malmasi and Dras for answering questions about their LDA classifier.

References

Daniel Blanchard, Joel Tetreault, Derrick Higgins, Aoife Cahill, and Martin Chodorow. 2013. TOEFL11: A corpus of non-native English. *ETS Research Report Series*, 2013(2):i–15.

Julian Brooke and Graeme Hirst. 2013. Using other learner corpora in the 2013 NLI shared task. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 188–196, Atlanta, Georgia. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *CoRR*, abs/2005.14165.

Andrea Cimino and Felice Dell'Orletta. 2017. Stacked sentence-document classifier approach for improving native language identification. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 430–437, Copenhagen, Denmark. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Gili Goldin, Ella Rabinovich, and Shuly Wintner. 2018. Native language identification with user generated content. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3591–3601, Brussels, Belgium. Association for Computational Linguistics.

Thomas G. Hudson and Sardar Jaf. 2018. On the development of a large scale corpus for native language identification. In *Proceedings of the 17th International Workshop on Treebanks and Linguistic Theories*, pages 115–129, Oslo, Norway. Linköping University Electronic Press.

Radu Tudor Ionescu and Marius Popescu. 2017. Can string kernels pass the test of time in native language identification? In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 224–234, Copenhagen, Denmark. Association for Computational Linguistics.

Radu Tudor Ionescu, Marius Popescu, and Aoife Cahill. 2014. Can characters reveal your native language? A language-independent approach to native language identification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1363–1373, Doha, Qatar. Association for Computational Linguistics.

Pavel Ircing, Jan Švec, Zbyněk Zajíc, Barbora Hladká, and Martin Holub. 2017. Combining textual and speech features in the NLI task using state-of-the-art machine learning techniques. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 198–209, Copenhagen, Denmark. Association for Computational Linguistics.

Moshe Koppel, Jonathan Schler, and Kfir Zigdon. 2005. Determining an author's native language by mining a text for errors. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 624–628, Chicago, Illinois, USA. ACM.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *Eighth International Conference on Learning Representations*, Addis Ababa, Ethiopia. ICLR.

Maryellen C. MacDonald. 2013. How language production shapes language form and comprehension. *Frontiers in Psychology*, 4:226.

Shervin Malmasi and Mark Dras. 2018. Native language identification with classifier stacking and ensembles. *Computational Linguistics*, 44(3):403–446.

Shervin Malmasi, Keelan Evanini, Aoife Cahill, Joel Tetreault, Robert Pugh, Christopher Hamill, Diane Napolitano, and Yao Qian. 2017. A report on the 2017 native language identification shared task. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 62–75, Copenhagen, Denmark. Association for Computational Linguistics.

Ilia Markov, Vivi Nastase, and Carlo Strapparava. 2018a. Punctuation as native language interference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3456–3466, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Ilia Markov, Vivi Nastase, Carlo Strapparava, and Grigori Sidorov. 2018b. The role of emotions in native language identification. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 123–129, Brussels, Belgium. Association for Computational Linguistics.

Ella Rabinovich, Yulia Tsvetkov, and Shuly Wintner. 2018. Native language cognate effects on second language lexical choice. *CoRR*, abs/1805.09590.

Björn W. Schuller, Stefan Steidl, Anton Batliner, Julia Hirschberg, Judee K. Burgoon, Alice Baird, Aaron C. Elkins, Yue Zhang, Eduardo Coutinho, and Keelan Evanini. 2016. The INTERSPEECH 2016 computational paralinguistics challenge: Deception, sincerity & native language. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 2001–2005, San Francisco, California, USA. ISCA.

Philipp Singer, Fabian Flöck, Clemens Meinhart, Elias Zeitfogel, and Markus Strohmaier. 2014. Evolution of Reddit: From the front page of the internet to a self-referential community? In *Proceedings of the 23rd International Conference on World Wide Web*, pages 517–522, Seoul, Korea. ACM.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. ERNIE 2.0: A continual pre-training framework for language understanding. In *34th AAAI Conference on Artificial Intelligence*, pages 8968–8975, New York, New York, USA. AAAI.

Joel Tetreault, Daniel Blanchard, and Aoife Cahill. 2013. A report on the first native language identification shared task. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 48–57, Atlanta, Georgia. Association for Computational Linguistics.

Joel Tetreault, Daniel Blanchard, Aoife Cahill, and Martin Chodorow. 2012. Native tongues, lost and found: Resources and empirical evaluations in native language identification. In *Proceedings of COLING 2012*, pages 2585–2602, Mumbai, India. The COLING 2012 Organizing Committee.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS 2017)*, pages 5998–6008, Long Beach, California, USA. NeurIPS.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big Bird: Transformers for longer sequences. *CoRR*, abs/2007.14062.

Does a Hybrid Neural Network based Feature Selection Model Improve Text Classification?

Suman Dowlagar
LTRC
IIIT-Hyderabad
suman.dowlagar@
research.iiit.ac.in

Radhika Mamidi
LTRC
IIIT-Hyderabad
radhika.mamidi@
iiit.ac.in

Abstract

Text classification is a fundamental problem in the field of natural language processing. Text classification mainly focuses on giving more importance to all the relevant features that help classify the textual data. Apart from these, the text can have redundant or highly correlated features. These features increase the complexity of the classification algorithm. Thus, many dimensionality reduction methods were proposed with the traditional machine learning classifiers. The use of dimensionality reduction methods with machine learning classifiers has achieved good results. In this paper, we propose a hybrid feature selection method for obtaining relevant features by combining various filter-based feature selection methods and fastText classifier. We then present three ways of implementing a feature selection and neural network pipeline. We observed a reduction in training time when feature selection methods are used along with neural networks. We also observed a slight increase in accuracy on some datasets.

1 Introduction

Text classification assigns one or more class labels from a predefined set to a document based on its content. Text classification has broad applications in real-world scenarios such as document categorization, news filtering, spam detection, Optical character recognition (OCR), and intent recognition. Giving high weights to relevant features is the objective of text classification.

The field of text classification has gained more interest during the machine learning (ML) era. Many discriminative and generative machine learning classifiers have achieved excellent results in the field of text classification (Deng et al., 2019). Feature selection and feature extraction methods are often used to reduce high dimensionality (Bharti and Singh, 2015). Feature extraction generates features from text (Agarwal and Mittal, 2014). Feature selection (FS) selects the most prominent features (Saleh and El-Sonbaty, 2007).

These feature selection and extraction methods are used along with traditional classification algorithms. These methods reduced the curse of dimensionality and increased the classification accuracy (Deng et al., 2019).

Recently, deep learning models are used to learn better text representations and to classify the text (Minaee et al., 2020). Such models include convolutional neural networks (CNN) (Kim, 2014), recurrent neural networks (RNN) (Hochreiter and Schmidhuber, 1997), Transformer models (Adhikari et al., 2019), and graph convolutional networks (GCN) (Yao et al., 2019). These NN models capture semantic and syntactic information in local and global word sequences.

Even though the neural networks capture a complex and dense representation of data, the set of words introducing noise in the classifier is still present. Such words add the burden of increased vocabulary, which results in increased textual representation and an increase in the training time of the classifiers (Song et al., 2011).

Similar to the traditional approaches, we want to understand the effects of using statistical feature selection algorithms beforehand to calculate the features' relevance and then train a fastText text-classification algorithm on those relevant features. Using this feature selection and neural network pipeline, we assume that the complexity of dealing with larger vocabulary decreases. Including feature selection with fastText text-classification helps reduce the classifier's training time and helps the classifier reach better local optima, showing a significant increase in classification accuracy.

In this work, we analyzed a feature selection and neural network pipeline for text classification.

Proceedings of the 17th International Conference on Natural Language Processing, pages 272–280
Patna, India, December 18 - 21, 2020.

We used a hybrid feature selection method to get a score on relevant features. Using this score, we formulated three methods. The first and second methods deal with modifying the original text by extracting the relevant features. The third method deals with using the feature selection scores and pass it along with the word embeddings. We then observed the effect of feature selection on various neural networks.

The rest of the paper is organized as follows. Section 2 gives a brief review of previous works in the field of feature selection and text classification. Section 3 presents a detailed procedure of the proposed pipeline and presents the experiments and datasets used for our study. Section 4 reports the performance of text classifiers with and without feature selection methods. Section 5 concludes the paper.

2 Literature Survey

This section presents a brief description of the neural network (NN) classification algorithms and various feature selection methods.

2.1 Deep Learning for Text Classification

Nowadays, various NNs such as CNN, RNN, BERT, and Text GCN achieve state-of-the-art results on text classification. CNN uses 1d convolutions (Zhang et al., 2015) and character level convolutions (Conneau et al., 2016) to learn the semantic similarity of words or characters, which helps in classifying the text. RNN models such as GRU, LSTM, and BiLSTM (Liu et al., 2016) take word to word sequences to learn a better textual representation of a document that helps in text classification. Attention mechanisms have been introduced in these LSTM models, which increased the representativeness of the text for better classification (Yang et al., 2016). Transformer models such as BERT (Devlin et al., 2018) uses the attention mechanism that learns contextual relations between words or sub-words in a text (Adhikari et al., 2019). Text GCN (Yao et al., 2019) uses a graph-convolutional network to learn a heterogeneous word document graph on the whole corpus. Text GCN can capture global word co-occurrence information and use graph convolutions to learn a global representation, which helps classify the documents.

2.2 Feature Selection on Text Data

The text classification often involves extensive data with thousands of features. Although tens of thousands of words are in a typical text collection, most of them contain little or no information to predict the text label. These features introduce complexity and increase the training time of an ML classifier. Feature selection is one method for giving high scores to relevant features (Deng et al., 2019). The goal of feature selection is to select highly-relevant features with minimum redundancy. The relevance of a feature indicates that the feature is always necessary to predict the class label.

There are various text feature selection methods in the literature, each being filter, wrapper, hybrid, and embedded methods. The filter method evaluates the quality of a feature using a scoring function. Some filter methods evaluate the goodness of a term based on how frequently it appears in a text corpus. Document Frequency (DF) (Lam and Lee, 1999) and Term Frequency - Inverse Document Frequency (TFIDF) (Rajaraman and Ullman, 2011) comes under this category. Other filter methods that originate from information theory are, Mutual Information (MI) (Taira and Haruno, 1999; Tang et al., 2019), Information Gain (IG) (Yang and Pedersen, 1997), CHI (Rogati and Yang, 2002), ANOVA F measure (Elssied et al., 2014), Bi-Normal Separation (BNS) (Forman, 2003) and the GINI method (Shang et al., 2013). They use hypothesis testing, contingency tables, mean and variance scores, conditional and posterior probabilities for selecting the features.

The wrapper method (Maldonado and Weber, 2009) use a search strategy to construct each possible subset, feeds each subset to the chosen classifier, and then evaluates the classifier's performance. These two steps are repeated until the desired quality of the feature subset is reached. The wrapper approach achieves better classification accuracy than filter methods. However, the time taken by the wrapper method is very high when compared to filter methods.

Embedded methods complete the FS process within the construction of the machine learning algorithm itself. In other words, they perform feature selection during the model training. An embedded method is Decision Tree (DT) (Quinlan, 1986). In DT, while constructing the classifier, DT selects the best features/attributes that may give the best discriminative power.

Hybrid methods are robust and take less time when compared to the wrapper and embedded methods. They combine a filter method with a wrapper method during the feature selection process. The HYBRID model (Günal, 2012) employs a combination of filter methods to select to rank the features and then a wrapper method to the obtained final features set. Our FS method is similar to the HYBRID model.

A detailed report on the benefits of using the feature selection methods in the pipeline with traditional classifiers is presented in Deng et al. (2019); Forman (2003).

Apart from using traditional classification methods, deep feature selection using neural networks were also proposed. These models use deep neural network autoencoders for the feature set reduction and text generation (Mirzaei et al., 2019; Han et al., 2018).

Lam and Lee (1999) studies the effect of feature set reduction before applying the neural network classifiers. The paper uses a multi-layer perceptron (MLP) classifier in combination with filter-based FS method. Alkhatib et al. (2017) proposes the use of neural network-based feature selection and text classification. Our work comes under this category.

3 Proposed Pipeline

In this section, we present our feature selection and neural network pipeline.

The feature selection and neural network pipeline start with selecting a good tokenizer to tokenize the data and create a feature set. The tokenizer used for our feature selection is the Sentencepiece tokenizer (Kudo and Richardson, 2018). Sentencepiece tokenizer implements subword units by using byte-pair-encoding (BPE) (Sennrich et al., 2015) and unigram language model (Kudo, 2018). In the feature subset generation, we considered a hybrid feature selection method known as HYBRID (Günal, 2012). It has proved that a combination of the features selected by various methods is more effective and computationally faster than the features selected by individual filter and wrapper methods. Similar to the HYBRID model, we used three filters to obtain the relevancy score. The filters we considered were CHI2, ANOVA-F, and MI. These filters calculate the relevancy between the word and the class labels.

Before feature selection, we used the Bag-of-Words(BoW) model to vectorize the data. In the BoW model, each feature vector is represented by TF_IDF scores.

Then we used statistical measures such as χ^2, ANOVA-F, and MI for obtaining feature scores.

χ^2 [1] is a statistic to measure a relationship between feature vectors and a label vector.

Analysis of Variance (ANOVA[2]) is a statistical method used to check the means of two or more groups that are significantly different from each other.

Mutual Information (MI[3]) is frequently used to measure the mutual dependency between two variables.

Using different statistical methods, we measured the relevance of each feature. We then aggregate the relevance scores of all satistical methods for each feature. The relevance of a feature x_i is given by,

$$Relevancy(x_i) = \chi^2(x_i) + ANOVA(x_i) + MI(x_i) \quad (1)$$

Instead of an LR classifier given in the HYBRID model, we used the fastText classifier (Joulin et al., 2016) for the feature selection. We used the fastText classifier as it is often on par with deep learning classifiers in terms of accuracy and performs faster computations. The fastText classifier treats the average of word embeddings as document embeddings, then feeds document embeddings into a feed-forward NN or a multinomial LR classifier. We used pre-trained fastText word embeddings (Grave et al., 2018) while training a classifier.

To get the final features list, we sorted the normalized, aggregated value in descending order and divided the entire feature space into k sets. In our model, we divided the sorted feature space into 20 sets. The value of k is fixed to 20 using a trial and error basis. We take the first set as the vocabulary of the classifier. We then trained the classifier and noted its accuracy. In the second iteration, we considered the vocabulary as the combination of first

[1] A detailed explanation and a simple example of χ^2 is given at https://www.mathsisfun.com/data/chi-square-test.html

[2] A detailed explanation for ANOVA is given in https://towardsdatascience.com/anova-for-feature-selection-in-machine-learning-d9305e228476.

[3] A simple explanation and working python example of MI is available at https://machinelearningmastery.com/information-gain-and-mutual-information/

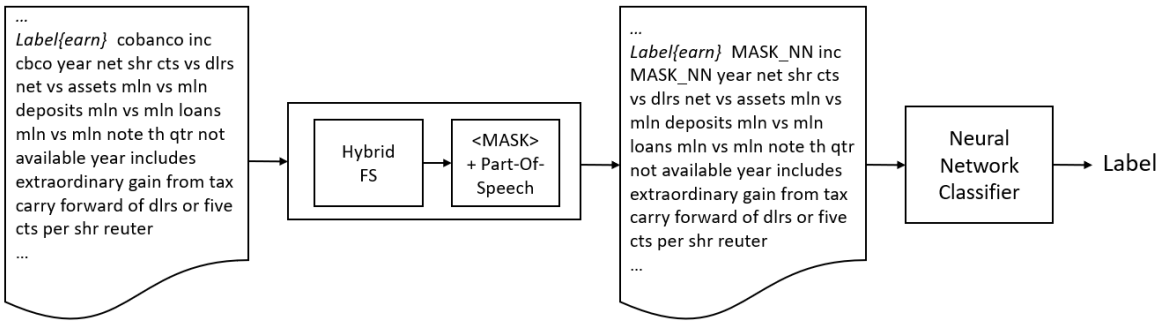


Figure 1: Modifying text by masking the low ranked words

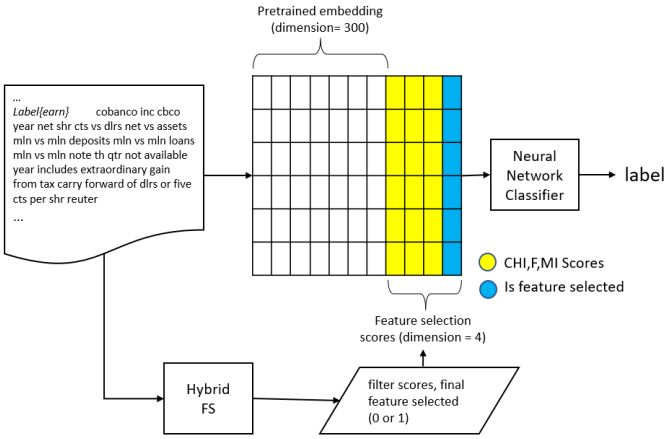


Figure 2: Meta-Embeddings, including feature scores along with word embeddings

and second sets. Similarly, the third set has the vocabulary of the first three sets combined. We repeated the process until all the lists are exhausted. The set of features that resulted in a better classification metric is considered as the final feature set.

According to the proposed FS method, the final feature set is considered relevant, and they are necessary to perform the text classification. In contrast, the other features have little to no effect on the text classification or might degrade the classifier's performance.

After feature subset generation, we propose three methods for including the feature selection information before training the neural network classifiers.

1. **Method 1 (Selecting only the relevant features)**[4]**:** Like traditional classification algorithms, we select only the relevant features that are estimated to be important by the feature selection method before training a neural network classifier.

2. **Method 2 (Masking the features that were given low importance by our FS method):** We felt that removing the features given low rank by our FS method might disturb the original data's grammatical structure, thus disturbing the word to word dependencies. We masked the low ranked words with the help of $< MASK > +POS(word)$ tag. $< MASK >$ word masks the low ranked word, and POS preserves the word's part of speech. The visual representation of method 2 is shown in figure 1

3. **Method 3 (Meta Embeddings):** As shown in figure 2, we pass the relevancy and feature selection information along with embeddings in this method. Each slot holds the filter scores, i.e., CHI, ANOVA, MI scores of each feature. The last slot holds a 1 or 0 value. 1 is used for the selected features, and 0 is used for low ranked features that were not selected by our hybrid feature selection approach.

We analyzed and evaluated the above methods with various state-of-the-art NN classifiers on the benchmark datasets.

3.1 Experiment

In this section, we evaluated our feature selection and neural network pipeline on two tasks. We wanted to determine:

- If the pipeline decreases the training time of the classifier
- If it helps in obtaining better local optima, thus improving the classification accuracy.

We tested our pipeline across multiple state-of-the-art text classification algorithms.

[4]This method is already used while selecting the final features set by the fastText classifier.

1. **CNN:** (Kim, 2014) This convolutional neural network-based text classifier is trained by considering pre-trained word vectors.

2. **Bi-LSTM:** (Liu et al., 2016) A two-layer, bi-directional LSTM text classifier with pre-trained word embeddings as input was considered for the task of text classification.

3. **fastText:** (Joulin et al., 2016) This is a simple, efficient, and the fastest text classification method. It treats the average of word/n-grams embeddings as document embeddings, then feeds document embeddings into a linear classifier.

4. **Text GCN:** (Yao et al., 2019) Builds a heterogeneous word document graph for a whole corpus and turns document classification into a node classification problem. It uses GCN (Kipf and Welling, 2017) to learn word and document embeddings.

5. **DocBERT:** (Adhikari et al., 2019) A fine-tuned BERT model for document classification. The BERT model (Devlin et al., 2018) uses a series of multiheaded attention and feedforward networks for various NLP tasks.

3.2 Datasets

We ran our experiments on three widely used benchmark corpora and multilingual corpora. They are 20Newsgroups(20NG), R8, and R52 of Reuters 21578 and MLMRD.

- The 20NG dataset contains 18,846 documents divided into 20 different categories. 11,314 documents were used for training, and 7,532 documents were used for testing.

- R52 and R8 are two subsets of the Reuters 21578 dataset. R8 has 8 categories of the top eight document classes. It was split into 5,485 training and 2,189 test documents. R52 has 52 categories and was split into 6,532 training and 2,568 test documents.

- MLMRD is a Multilingual Movie Review Dataset. It consists of the genre and synopsis of movies across multiple languages, namely Hindi, Telugu, Tamil, Malayalam, Korean, French, and Japanese. The data set is minimal and unbalanced. It has 9 classes and a total of 14,997 documents. The data was split into 10,493 training and 4,504 test documents.

We first preprocessed all the datasets by cleaning and tokenizing. The tokenizer used is the fastText tokenizer.

For baseline 1 models, we used multilingual fastText embeddings (Grave et al., 2018) of dimensionality 300, and baseline 2 models had the dimensionality of 304. We used default parameter settings as in their original papers for implementations. For calculating TFIDF, CHI2, ANOVA-F, MI scores, we used the scikit-learn library (Pedregosa et al., 2011). For POS tagging, we used the NLTK (Bird et al., 2009) pos tagger.

All the neural network models were run on the GPU processor on the Windows platform with NVIDIA RTX 2070 graphics card.

4 Performance

Datasets	Our FS	HYBRID FS
20Newsgroups	**81.27%**	77.34%
R8	**96.94%**	93.79%
R52	**92.72%**	86.43%
MLMRD	**47.09%**	42.98%

Table 1: The classification accuracy of our FS model when compared to the HYBRID model.

In our work, we modified the HYBRID (2012) feature selection model by changing the LR classifier to the fastText classifier. We selected the fastText classifier in the feature selection process because of its fast learning ability of a NN model compared to the traditional ML classifiers and other neural network classifiers (Joulin et al., 2016) without any decrease in classification accuracy. The neural network classifiers such as MLP, CNN, RNN, transformer, and GCN models achieve better classification accuracy when compared to traditional ML classifiers, but their training time is very high.

Using a fastText classifier during feature selection, we observed that our model performed better on all the benchmark datasets than the HYBRID model. The results are shown in table 1. The fastText classifier's use helped the model obtain better relevant features, increasing the current feature selection model's accuracy compared to the HYBRID model.

As mentioned above, we used the training time-taken and test accuracy as the metrics to evaluate our approach. The accuracy and training time are recorded by running the model 10 times, and the

Datasets	20Newsgroups	R8	R52	MLMRD
Baseline 1 & 2	1,01,631 (V)	19,956 (V)	26287 (V)	94073 (V)
Method 1	25732 (0.25V)	17364 (0.87V)	22372 (0.85V)	52015 (0.55V)
Method 2	25732+30 (0.25V)	17364+30 (0.87V)	22372+30 (0.85V)	52015+143 (0.55V)
Method 3	1,01,631 (V)	19,956 (V)	26287 (V)	94073 (V)

Table 2: The vocabulary size in all the FS inclusion methods when compared to the baselines. "V" is denoted as the vocabulary size of the actual data. Baselines 1,2, and method 3 have no change in vocabulary. However, using our FS method, the vocabulary is reduced to a maximum of 75% (for 20Newsgroups data). Other datasets have seen a 13% to 45% decrease in vocabulary size. We can see an increase in vocabulary from method 1 to method 2. It is due to the additional vocabulary resulted from the mask words when they are accompanied by pos tags. Here Penn Treebank POS tagset is used.

Datasets	Method	Classifier(s)				
		CNN	**Bi-LSTM**	**fastText**	**DocBERT**	**Text GCN**
20Newsgroups	Baseline 1	79.31%	73.60%	81.04%	**90.19%**	86.13%
	Baseline 2	79.46%	74.25%	82.44%	NA	86.23%
	Method 1	78.27%	73.44%	81.27%	89.37%	**86.25%**
	Method 2	77.29%	70.48%	80.14%	88.43%	85.65%
	Method 3	**80.59%**	**76.57%**	**84.48%**	NA	86.15%
R8	Baseline 1	97.24%	92.70%	96.13%	**97.62%**	96.80%
	Baseline 2	97.37%	93.82%	96.50%	NA	**96.94%**
	Method 1	**97.39%**	93.74%	96.94%	97.44%	96.28%
	Method 2	96.57%	94.34%	96.07%	97.44%	96.85%
	Method 3	**97.39%**	**96.74%**	**97.18%**	NA	**96.94%**
R52	Baseline 1	94.78%	87.53%	92.02%	92.95%	93.56%
	Baseline 2	**94.84%**	90.79%	92.76%	NA	93.64%
	Method 1	94.29%	87.47%	92.72%	**93.10%**	92.97%
	Method 2	91.71%	**91.90%**	90.30%	92.10%	93.19%
	Method 3	**94.84%**	91.48%	**92.83%**	NA	**93.74%**
MLMRD	Baseline 1	47.63%	46.43%	46.92%	**53.11%**	47.62%
	Baseline 2	47.79%	47.43%	48.92%	NA	49.62%
	Method 1	44.98%	44.82%	47.09%	51.90%	46.58%
	Method 2	44.63%	44.05%	46.61%	50.90%	46.98%
	Method 3	**48.44%**	**49.13%**	**49.55%**	NA	**51.50%**

Table 3: Test accuracy on various neural network classifiers for the task of document classification. As the BERT model used is a fine-tuned one, we did not modify the model.

average of the metrics was presented.

4.1 Effects of our methods on classification accuracy

Table 3 demonstrates the accuracy of feature selection methods on NN classifiers.

When methods 1 and 2 were used, there is a slight decrease in classification accuracy because the first two methods lost semantic connection among words. Thus, the classification performance is degraded. Also, some words which were relevant to the classifier were masked out during the FS method. Whereas in method 3, including the feature selection scores with word-embeddings, has shown a significant improvement in accuracy on all the datasets.

Compared to the other datasets, the 20NG dataset has seen a significant decrease in vocabulary size. The vocabulary was decreased by 75%. However, eliminating those features did not affect the accuracy of the classifier for methods 1 and 2.

Introducing the masked features in method 2 shown an increase in accuracy only in the Bi-LSTM method as this method considers word dependencies while training a classifier.

Including the feature selection scores along with

the word-embeddings improved the classification accuracy on all the datasets. The feature selection metadata helped the neural network classifier learn a better relationship between the words and classes and improve the classifier's accuracy by reaching better local optima.

In R8 and R52 datasets, we have seen an increase in accuracy using method 1 because our hybrid FS method worked better on these datasets by removing the noisy words without disturbing the relevant words. The maximum improvement in accuracy is shown in the R8 dataset, with a +4% increase in classification accuracy.

Our approach did not show any better results on MLMRD datasets as this dataset has a limited number of documents to train and test the data for some languages (Telugu, Tamil, Malayalam, Korean). Reducing vocabulary size by the FS method decreased the classification accuracy.

4.2 Effects of our methods on training time

The pictorial representation of time taken by the classifiers for all the datasets is given in appendix B of the supplementary material.

The time taken by method 1 is lower than in all baseline models. In method 1, as the text is modified by considering only relevant features, the vocabulary size is reduced, and the sentence length is reduced. It resulted in the more accelerated training of the neural network.

The time taken by baseline 2 and method 3 is similar because of the same embedding dimensionality of 304, but method 3 has achieved local optima a few epochs before compared to baseline 2, resulting in a time decrease of a few seconds. This phenomenon is attributed to the use of feature selection scores along with word embeddings.

Method 2 has shown an increase in training time even though the vocabulary is decreased because of 2 factors.

1. The masking of features created unknown words in the data, and the classifier has to be trained to learn the representation of masked words, whereas the other words had pre-trained embeddings.

2. Apart from vocabulary, the neural network training time also depends on the input batch size given to the network and the length of the sentence in each batch. Because of the masked words, there is no decrease in either batch size or the sentence length. So the masking of data did not decrease the training time of the classifier.

On the contrary, the Text GCN model has shown a decrease in training time because the classifier computes heterogeneous graph embeddings of each word based on the textual data before classification. It did not use any pre-trained embeddings.

In method 3, there is a slight increase in training time because of increased vocabulary size due to the inclusion of feature selection metadata.

Of all the NN classifiers, the Text GCN model had shown a maximum decrease in training time by 488 sec when method 1 was used on 20NG data. As the Text GCN operates on building a graph on the complete vocabulary of data, the time taken by the method to build the graph is reduced significantly by reducing the vocabulary size. It is followed by the DocBERT and Bi-LSTM on 20NG data with a decrease in training time by 480 and 394 sec. Text GCN and Bi-LSTM have shown a significant decrease in training time on all the datasets. On the contrary, fastText and CNN are very fast while training the NN model. The training time of such models was unchanged when our method 1 was used.

When compared to all the classifiers, DocBERT achieved better results because of its evolutionary multi-headed attention and transformer models. As the Text GCN captures both local and word embeddings by constructing a heterogeneous graph, their results were better than those of the CNN and Bi-LSTM models, which work only on local word dependencies. As we increased the size of the embedding in FS method 2, this increased the dimensionality of vocabulary, resulting in the classifier's increased training time.

5 Conclusion

In our work, "Does a Hybrid NN FS Model Improve Text Classification?", we used the NN based hybrid FS method to extract relevant features and used NN classifiers for text classification. We extracted the relevant or high ranked features using filter-based methods and a fastText classifier. We then proposed three methods on how the feature selection can be included in the NN classification process. First, modifying the corpus by considering only relevant features. Second, modifying the data by masking the low ranked features, and

the third method introduces feature selection information along with word embeddings. We observed that method 1 had shown a significant reduction in training time when large datasets or slower models are used, accompanied by a minimal change in classification accuracy. By introducing $MASK + P0S(word)$, we inferred that the masked word was a burden to the classifier, and it always tried to adjust the word embeddings, which resulted in increased epoch time during training and a slightly negative effect on classification accuracy. Whereas method 3 has shown no effect on decreasing the training time, it has shown a maximum of 4% increase in the classification accuracy compared to baseline. It proved that introducing feature scores along with pre-trained word embeddings while training the NN classifier is beneficial.

Instead of opting for random naive vocabulary reduction techniques such as using min_df and max_df (minimum and maximum document frequency) for selecting features, by using FS methods, we can calculate the relevance of the word beforehand and use that as metadata to the NN classifier. When the datasets are huge, these methods are of more significance. We can use the modified data while tuning the hyperparameters. Then we can use the real data to train and evaluate the model. Even in the critical domain datasets such as "medical", we cannot rely on removing a word based on min_df and max_df scores. Each word in those datasets should be treated with utmost significance. FS methods help in such scenarios by calculating the word's relevance and helps maintain better vocabulary before training neural network classifiers.

References

Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019. Docbert: Bert for document classification. *arXiv preprint arXiv:1904.08398*.

Basant Agarwal and Namita Mittal. 2014. Text classification using machine learning methods-a survey. In *Proceedings of the Second International Conference on Soft Computing for Problem Solving (SocProS 2012), December 28-30, 2012*, pages 701–709, New Delhi. Springer India.

Wael Alkhatib, Christoph Rensing, and Johannes Silberbauer. 2017. Multi-label text classification using semantic features and dimensionality reduction with autoencoders. In *International Conference on Language, Data and Knowledge*, pages 380–394. Springer.

Kusum Kumari Bharti and Pramod Kumar Singh. 2015. Hybrid dimension reduction by integrating feature selection with feature extraction method for text clustering. *Expert Systems with Applications*, 42(6):3105–3114.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*.

Xuelian Deng, Yuqing Li, Jian Weng, and Jilian Zhang. 2019. Feature selection for text classification: A review. *Multimedia Tools and Applications*, 78(3):3797–3816.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Nadir Omer Fadl Elssied, Othman Ibrahim, and Ahmed Hamza Osman. 2014. A novel feature selection based on one-way anova f-test for e-mail spam classification. *Research Journal of Applied Sciences, Engineering and Technology*, 7(3):625–638.

George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. *Journal of machine learning research*, 3(Mar):1289–1305.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. *arXiv preprint arXiv:1802.06893*.

Serkan Günal. 2012. Hybrid feature selection for text classification. *Turkish Journal of Electrical Engineering and Computer Science*, 20(Sup. 2):1296–1311.

Kai Han, Yunhe Wang, Chao Zhang, Chao Li, and Chao Xu. 2018. Autoencoder inspired unsupervised feature selection. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2941–2945. IEEE.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

TN Kipf and M Welling. 2017. Semi-supervised classification with graph convolutional networks iclr.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Savio LY Lam and Dik Lun Lee. 1999. Feature reduction for neural network based text categorization. In *Proceedings. 6th international conference on advanced systems for advanced applications*, pages 195–202. IEEE.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

Sebastián Maldonado and Richard Weber. 2009. A wrapper method for feature selection using support vector machines. *Information Sciences*, 179(13):2208–2217.

Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2020. Deep learning based text classification: A comprehensive review. *arXiv preprint arXiv:2004.03705*.

Ali Mirzaei, Vahid Pourahmadi, Mehran Soltani, and Hamid Sheikhzadeh. 2019. Deep feature selection using a teacher-student network. *Neurocomputing*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

J. Ross Quinlan. 1986. Induction of decision trees. *Machine learning*, 1(1):81–106.

Anand Rajaraman and Jeffrey David Ullman. 2011. *Mining of massive datasets*. Cambridge University Press.

Monica Rogati and Yiming Yang. 2002. High-performing feature selection for text classification. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, CIKM '02, page 659–661, New York, NY, USA. Association for Computing Machinery.

S. N. Saleh and Y. El-Sonbaty. 2007. A feature selection algorithm with redundancy reduction for text classification. In *2007 22nd international symposium on computer and information sciences*, pages 1–6.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Changxing Shang, Min Li, Shengzhong Feng, Qingshan Jiang, and Jianping Fan. 2013. Feature selection via maximizing global information gain for text classification. *Knowledge-Based Systems*, 54:298–309.

Qinbao Song, Jingjie Ni, and Guangtao Wang. 2011. A fast clustering-based feature subset selection algorithm for high-dimensional data. *IEEE transactions on knowledge and data engineering*, 25(1):1–14.

Hirotoshi Taira and Masahiko Haruno. 1999. Feature selection in svm text categorization. In *AAAI/IAAI*, pages 480–486.

Xiaochuan Tang, Yuanshun Dai, and Yanping Xiang. 2019. Feature selection based on feature interactions with application to text categorization. *Expert Systems with Applications*, 120:207–216.

Yiming Yang and Jan O Pedersen. 1997. A comparative study on feature selection in text categorization. In *Icml*, volume 97, page 35.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7370–7377.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Efforts Towards Developing a Tamang Nepali Machine Translation System

Binaya K. Chaudhary[1] **Bal Krishna Bal**[2] **Rasil Baidar**[1]

Information and Language Processing Research Lab
Department of Computer Science and Engineering
Kathmandu University
Dhulikhel, Kavre, Nepal
[1]{binayachaudari, rasilgrt}@gmail.com
[2]bal@ku.edu.np

Abstract

The Tamang language is spoken mainly in Nepal, Sikkim, West Bengal, some parts of Assam, and the North East region of India. As per the 2011 census conducted by the Nepal Government, there are about 1.35 million Tamang speakers in Nepal itself. In this regard, a Machine Translation System for Tamang-Nepali language pair is significant both from research and practical outcomes in terms of enabling communication between the Tamang and the Nepali communities. In this work, we train the Transformer Neural Machine Translation (NMT) architecture with attention using a small hand-labeled or aligned Tamang-Nepali corpus (15K sentence pairs). Our preliminary results show BLEU scores of **27.74** for the Nepali→Tamang direction and **23.74** in the Tamang→Nepali direction. We are currently working on increasing the datasets as well as improving the model to obtain better BLEU scores.

1 Introduction

Machine Translation (MT) Systems today represent one of the biggest achievements of human mankind in the field of Artificial Intelligence (AI) and Language Technologies (LT). With the advancement of Deep Neural Networks (DNN) and abundance of data, today's MT Systems already claim more than 90% accuracy in the translation thus opening doors for the adoption and use of highly reliable translation systems.

These systems serve multiple folds: (1) bridge the language barrier between different language speaking communities; (2) preserve languages that are not as popularly spoken or used in the younger generations and many more. If we talk about the Tamang language, native to the Tibeto-Burman group of the Sino-Tibetan language family and spoken by about 1.35 million native speakers in Nepal according to the 2011 census by the Nepal Government[1], then we can say that it can greatly benefit from MT systems. This is further enunciated by the fact that Tamang is one of the languages which is highly affected by the latest migration trends, both within and outside the country, in quest of better lives and opportunities. Consequently, the language is not being spoken or learned by the younger generation and is being limited to the older generations thus creating a fear of extinction. On the contrary, in areas with dense Tamang populations in Nepal the primary language of communication is Tamang and even the medium of teaching is Tamang in these areas. Needless to note, MT Systems can open up the Tamang community from such areas to the outer world via the knowledge sources available in Nepali and English languages on the Internet.

In this paper, we discuss our efforts on developing a Tamang↔Nepali Machine Translation System. The biggest contribution of this work is the development of the parallel corpus of 15,000 parallel sentences in Tamang and Nepali from scratch. Moreover, we have adopted the Transformer architecture (Vaswani et al., 2017) for a low-resource language pair (Tamang-Nepali) with quite a few language-based customizations of hyper-parameters. We have achieved a BLEU score of **27.74** for Nepali→Tamang and a BLEU score of **23.74** for Tamang→Nepali on the test sets.

2 Challenges in Parallel Corpus Development

This is the first work of any kind in terms of Nepali↔Tamang Machine Translation; there is no related work in this regard. When we started working on the project, we did not have any parallel

[1] https://unstats.un.org/unsd/demographic-social/census/documents/Nepal/Nepal-Census-2011-Vol1.pdf

Proceedings of the 17th International Conference on Natural Language Processing, pages 281–286
Patna, India, December 18 - 21, 2020.

sentences for the Tamang-Nepali pair. Hence, we started to look for any available resources for the language pair. The problem with Tamang is that there is still no consensus in the community regarding the script for the language. There are a significant number of people in the Tamang community advocating for the Tamyig script which is very close to Tibetan script. However, since the Tamyig script is mostly confined to scriptures, holy books of the Lamas and not taught widely in schools, a large chunk of the Tamang community still uses the Devanagari script for writing and print media. Hence, we can find quite a few publications in Tamang written in the Devanagari script. Based on the popular usage and availability, we also decided to opt for Devanagari script-based Tamang language texts. Our very first preliminary attempt included seeing if we could use the Bible translations in Nepali[2] and Tamang[3] as the parallel sentences. However, the resource had its own set of issues

- the quality of the translation was not good,
- sentence-level alignment of translations did not yield satisfactory results,
- there was an issue of inconsistent use of the Eastern and Western Tamang, dialects of the Tamang language in Nepal.

So, we dropped the idea of using this resource. We also tried using Tamang songs and their translations but this attempt was not fruitful as the translations lacked wide coverage of texts of varying complexity in the languages. Finally, we established contact with Tamang linguists and experts as well as activists who had been working for the Tamang language in different capacities for long and we decided to initially develop 15,000 parallel sentences for Tamang-Nepali. These sentences range from simple to medium and complex sentences and have been taken from different sources like day-to-day spoken communication texts, child storybooks, general articles from Tamang language magazines, literary articles, etc.

[2] https://www.bible.com/bible/1711/MAT.1

[3] https://www.bible.com/bible/1177/MAT.1

3 Crowdsourcing the Development of Parallel Corpus

In order to make the process of the parallel corpus development more managed, we developed a tool, namely the Corpus Development Software which is a platform for submitting translations to the pre-selected sentences for the source language, as one sentence at a time. This approach prevents the misalignment of the sentences in due course of translation as the task demands one sentence in the target for each source sentence. The tool does not just provide an interface for submission of the translations but also lets us know the assigned translator regarding the deadline and what the status of the assigned task is ("Not Started", "In Progress", "Completed", "Past due" etc.). Once the translator submits the task, then the system similarly assigns the task to the reviewer and accordingly sets the status of the task for the review phase. Only after the reviewer submits the task, it becomes part of the finalized parallel corpus. We present the high-level workflow of the Corpus Development tool in Figure 1.

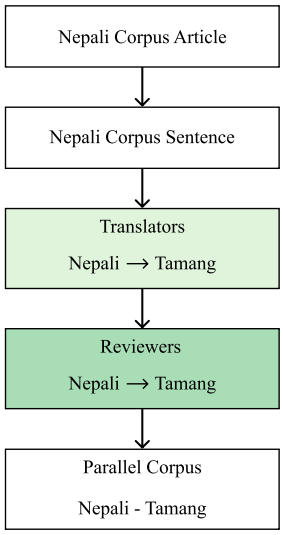


Figure 1: Workflow of the Corpus Development tool

4 Choosing the Right MT System Architecture

We studied a few state-of-the-art NMT architectures in due course of finalizing the MT architecture for our system. The first one included Google's Neural Machine Translation system (GNMT) (Wu et al., 2016), developed by Google. It is an LSTM network with 8 layers of encoders and decoders

with an attention mechanism. GNMT (Wu et al., 2016) tackles some of the well-known problems in NMT such as slower training, ineffectiveness in dealing with rare words, etc.

Similarly, Gehring et al. (2017) introduced a NMT architecture based on Convolutional Neural Network (CNN) for sequence-to-sequence learning that outperformed GNMT (Wu et al., 2016) on WMT'14 English to German translation and WMT'14 English-French.

Johnson et al. (2017) suggest a simple solution for a multilingual translation system using a single NMT model. The proposed model is identical to the GNMT system (Wu et al., 2016) with some optional connections on the actual network and few modifications in the input sequence. Johnson et al. (2017) describes the improvements in translation quality of low resourced language pairs when low-resourced language pairs and high resourced language pairs are mixed into a single model.

For this research, we follow the recent state-of-the-art Transformer (Vaswani et al., 2017) NMT architecture to create a fully supervised Neural Machine Translation system for developing a translation system for the Tamang-Nepali language pair. The core concept behind the Transformer model is self-attention—the ability to look at the multiple positions of an input sequence to understand/compute the representation of the sequence. This approach has proven to be better than recurrent methods, popularly being used for sequence-to-sequence learning. Transformer architecture addresses the recursion problem and allows parallelization, therefore reducing training time and increasing the performance making it cheaper and quicker to train. It can handle longer-range dependencies than most other translation models.

5 Experimental Setup

In this section, we present the experimental settings used for training the MT architecture and models for experimenting and reporting the results. The models which varied with different vocabulary of pre-selected words (1K, 2.5K and 5K) were trained using *fairseq*[4] toolkit (version 0.9.0) (Ott et al., 2019). Google Colab[5] (free-tier) is used for training the MT architectures with hardware acclerator set to GPU.

5.1 Experiment Settings

The Transformer model is a general sequence to sequence model with self-attention. We pass the input sentence through 5 layers of encoder stacked on top of each other that generates an output for each word/token in the sentence and 5 decoder layers to the encoder's output with its own input (self-attention) to predict the next word. The model handles variable-sized input using self-attention heads. We use 8 attention heads for both the encoder and the decoder. Similarly, the number of embedding dimensions and inner-layer dimensions used are 512 and 2048, respectively. We train the model with a learning rate of $7e{-}4$, the minimum learning rate being $1e{-}9$ for 150 epochs with a batch size of 64, updating a checkpoint after every 10 epochs. We set dropout, weight decay, and label smoothing to be 0.4, 10^{-4}, and 0.2 respectively. Adam optimizer with betas (0.9, 0.98) is used to optimize the model.

5.2 Data and Pre-processing

The corpus is developed from scratch accruing precise data directly from community linguists, which contains Nepali (Nep) and Tamang (Tamg) translation aligned at the sentence level; though the corpus is directly collected from community linguists, many instances of repeated sentence pairs were found, all such repeated sentence pairs are removed programmatically keeping alignment intact. After the removal of repeated sentence pairs, there are around 8243 + 3622 training sentence pairs, where 3622 sentence pairs are used for validation/development. We report results on the test set containing 3023 sentence pairs. The **Train**, **Test** and **Valid** dataset split is shown in Table 2.

	Sentence Pairs
Train	8243
Test	3023
Valid	3622

Table 2: Train, Test and Valid dataset split

IndicNLP[6] library (Kunchukuttan, 2020) is used to normalize and tokenize both Nepali and Tamang language texts. Translation, although being an open vocabulary problem, it is not possible to feed all the possible words in a language into a model.

[4] https://github.com/pytorch/fairseq
[5] https://colab.research.google.com/
[6] https://github.com/anoopkunchukuttan/indic_nlp_library

Translation Direction	Test	Valid	Vocabulary Size
Nepali→Tamang	**27.74**	29.41	1000
Tamang→Nepali	**23.74**	24.91	
Nepali→Tamang	27.33	29.07	2500
Tamang→Nepali	22.41	23.75	
Nepali→Tamang	26.13	27.53	5000
Tamang→Nepali	22.08	22.95	

Table 1: BLEU scores for Test and Valid set trained for 150 epochs using multiple vocabulary sizes.

Sennrich et al. (2015) describes the capability of open-vocabulary translation by encoding rare and unknown words as a sequence of subword units. For learning the vocabulary of source and target language, we use BPE[7] (Gage, 1994). BPE is the standard technique in Neural Machine Translation (NMT) and has been applied successfully in many systems. Ding et al. (2019); Gupta et al. (2019) argues that the impact of vocabulary size is significant in a low resourced dataset and that optimal BPE for Transformer architectures is small for low resource languages. Thus, based on the findings of Ding et al. (2019); Gupta et al. (2019), we chose the vocabulary size as $\leq$ 5000. Tokens not included in the vocabulary are replaced by universal tokens <unk>. Sentencepiece[8] library (Kudo and Richardson, 2018) is used to learn BPE in both the source and the target languages.

6 Results

After training all the models for 150 epochs, the best performing checkpoint[9] is used for each model with beam size of 5 (Yang et al., 2018) and length penalty of 1.2 to measure the translation quality using the BLEU[10] metric. The BLEU score (Papineni et al., 2002) is a general way to evaluate the performance of machine translation system when there can be multiple right outputs. Sacrebleu[11] (version 1.4.10) (Post, 2018) is used to compute the BLEU scores. BLEU scores obtained after training on multiple vocabulary sizes (1K, 2.5K and 5K) are shown in Table 1.

The model trained with vocabulary size of 1000 performs better among all and was able to obtain a BLEU score of **27.74** in Nepali→Tamang direction and **23.74** in Tamang→Nepali direction. Translation examples by the system generated by applying the best performing model are as follows:

Nepali→Tamang

Source	बच्चाहरुको निहुमा मुखमा आएजति शब्दमा अल्झेको मायाको आमा पनि घरको झ्यालमा बसेर दुईटी बच्चीलाई हेरिरहेकी थिईन् ।
Reference	कोलाकादेला निउरि सुङरि खातेदोना छिकरि हाल्बा मायाला आमा नोन दिमला झ्यालरि चिसि कोला ङिदान च्यासि चिबा मुबा ।
System	कोलाकादेला कुरि हापख्याइरि छिकरि आझोबा मायाला नोन दिमला झ्यालरि चिसि कोला ङहिदा च्याचिबा मुबा ।

Source	क्या महान् ईश्वर जसले थर्थरी कामेर संत्रस्त भएका अर्जुनमा शौर्य भर्दछ र कस्तो ठुलो योद्धा जो समाप्त भैसकेको शत्रुमाथि आक्रमण गर्दछ !
Reference	तिला ग्रेन ला थेसे लोङसि लगलग दार्बा अर्जुनरि पराक्रम युला ओम खाराङबा ग्रेन योद्धा जो जिनजिन्बा सत्तुरफिरि आक्रमण लाला !
System	क्या महान ग्लुसि निससे च्याङना गे लासि संत्रस्त ताबा अर्जुनरि शौर्य भर्दाम्ला ओम खाराङबा ठुलो चु जोद्धा चु जोप्त तासि जिन्बा ल्हुइरि आक्रमण लाला !

Source	त्यो इ.पू. २०० र सन् ३०० का बीचको अवधि– पनि चाखलाग्दो छ र यस अवधिका महत्वपूर्ण लक्षणहरू छन् ।
Reference	थे इ.पू. २०० थेन सन् ३०० ला गुङला दुइ– नोन चाखलाग्दो मुला ओम चु दुइला महत्वपूर्ण लक्षणजुगु मुला ।
System	थे इ . पू . २०० थेन सन् ३००० दुइला अवधि– नोन चालाबा मुला ओम चु अवधिला महत्वपूर्णजुगु लक्षणजुगु मुला ।

[7] Byte-Pair Encoding: A data compression technique

[8] https://github.com/google/sentencepiece

[9] Fairseq saves best performing checkpoint as "checkpoint_best.pt"

[10] BLEU: Bilingual Evaluation Understudy

[11] https://github.com/mjpost/sacreBLEU

Tamang→Nepali

Source	चुरि पाङ्बा मुतेबा अहंकार छ्याम चु वक्तव्यसे देन्बा दुइदा उन्बा मुला ।
Reference	यहाँ व्यक्त सम्पूर्ण अहंकार सहित यस वक्तव्यले वास्तविक अवस्थालाई देखाएको छ ।
System	यहाँ बोल्ने सबै अहंकार सबसँगै वक्तव्यले साँच्चालाई देखाएको छ ।

Source	शताब्दीयौं हेन्सेला गीताला अस्ङख्य थेन ल्होलो किसिमला फ्राल्तामन्हाङरि खाजिबाइ याखार्गिसे नोन चु मूलभूत रुपरेखादा अतिक्रमण लाबा आरे ।
Reference	शताब्दीयौंदेखिका गीताका असङ्ख्य र विविध किसिमका व्याख्यामध्ये कुनै एउटाले पनि यस मूलभूत रूपरेखालाई अतिक्रमण गरेको छैन ।
System	शताब्दीयौंदेखि गीताको अस्ख्य र विभिन्न किसिमको फ्याल्तमा आइसकेको एउटा मूलभूत रूपले पनि यो मूलभूत रूपलाई अतिक्रमण गर्ने छैन ।

Source	विडम्बनावस्, कम्युनिष्ट हिन्ना बिबा थेनोन जीवजुगुला चोथेबा जमात तिनि थोक हिसाबसे सढाउ फुमरि स्खलित ताजिबा मुला ।
Reference	विडम्बनावस्, कम्युनिष्ट हौं भन्ने उनै जीवहरूको यत्रो जमात आज थोक हिसाबले सढाउ अण्डामा स्खलित भइरहेको छ ।
System	विडम्बनावस् , कम्युनिष्ट हो भन्ने तिनै जीवनहरूको यत्र ज्ञान आज लिएर हिसाबले सढाउ फुत्तमा स्खलित भएका छन् ।

7 Discussion

For a low-resource language like Tamang, which has very negligible digital footprints, the achieved BLEU scores are quite encouraging. Besides, the gathered dataset (approx. 15K) can serve as a benchmark data for the Tamang - Nepali MT. We consider this as a significant contribution and achievement for the language pair. The deployed MT system is made available in the link[12] and is open for testing. We are really encouraged by the preliminary feedback that we have received from the community.

8 Future Plans and Road Map

We are in touch with the linguists and experts from the community and working towards further increasing the dataset. Similarly, we will also be looking further into how the MT model can be further optimized via hyper-parameter tuning, vocabulary size modification, changes in the learning rate etc.

The Tamang community has been involved in the testing of the system throughout and they are quite satisfied with the gradual improvement of the quality of the system. We plan to increase the size of the parallel sentences to 100K via crowdsourcing and other methods. We also plan to extend the bi-lingual MT system to a trilingual one incorporating English to the system so that the Tamang and the Nepali community benefit from the knowledge sources available in English on the Internet.

[12] https://translation.ilprl.ku.edu.np/

Acknowledgments

We would like to thank the **Information and Language Processing Research Lab, Kathmandu University** for providing the research opportunity and platform for this work. Similarly, our sincere thanks goes to Mr. Amrit Yonjan-Tamang and his team from Tamang Nangkhor for their support in the parallel corpus development. We would also like to extend our gratitude to Mr. Dawa Tamang, Virginia, US for providing support to the work.

References

Shuoyang Ding, Adithya Renduchintala, and Kevin Duh. 2019. A call for prudent choice of subword merge operations in neural machine translation. In *Proceedings of Machine Translation Summit XVII Volume 1: Research Track*, pages 204–213, Dublin, Ireland. European Association for Machine Translation.

Philip Gage. 1994. A new algorithm for data compression. *C Users J.*, 12(2):23–38.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning.

Rohit Gupta, Laurent Besacier, Marc Dymetman, and Matthias Gallé. 2019. Character-based nmt with transformer.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing.

Anoop Kunchukuttan. 2020. The IndicNLP Library. https://github.com/anoopkunchukuttan/

indic_nlp_library/blob/master/docs/indicnlp.pdf.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation.

Yilin Yang, Liang Huang, and Mingbo Ma. 2018. Breaking the beam search curse: A study of (re-)scoring methods and stopping criteria for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3054–3059, Brussels, Belgium. Association for Computational Linguistics.

Event Argument Extraction using Causal Knowledge Structures

Debanjana Kar[1], **Sudeshna Sarkar**[2], and **Pawan Goyal**[3]

Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur.
[1]debanjana.kar@iitkgp.ac.in
[2,3]{*sudeshna, pawang*}*@cse.iitkgp.ac.in*

Abstract

Event Argument extraction refers to the task of extracting structured information from unstructured text for a particular event of interest. The existing works exhibit poor capabilities to extract causal event arguments like Reason and After Effects. Furthermore, most of the existing works model this task at a sentence level, restricting the context to a local scope. While it may be effective for short spans of text, for longer bodies of text such as news articles, it has often been observed that the arguments for an event do not necessarily occur in the same sentence as that containing an event trigger. To tackle the issue of argument scattering across sentences, the use of global context becomes imperative in this task. In our work, we propose an external knowledge aided approach to infuse document level event information to aid the extraction of complex event arguments. We develop a causal network for our event-annotated dataset by extracting relevant event causal structures from ConceptNet and phrases from Wikipedia. We use the extracted event causal features in a bi-directional transformer encoder to effectively capture long-range inter-sentence dependencies. We report the effectiveness of our proposed approach through both qualitative and quantitative analysis. In this task, we establish our findings on an event annotated dataset in 5 Indian languages. This dataset adds further complexity to the task by labeling arguments of entity type (like Time, Place) as well as more complex argument types (like Reason, After-Effect). Our approach achieves state-of-the-art performance across all the five languages. Since our work does not rely on any language specific features, it can be easily extended to other languages as well.

1 Introduction

Event argument extraction is a key information extraction task that extracts structured information from unstructured texts. A widely studied task, it often involves multiple complex sub-tasks like entity retrieval, trigger detection and event-entity linking. In almost all the existing works, the terms event arguments and entities have been used interchangeably. Popular entity types which are frequently mined are that of place, time, organisations, and persons of interest. Our data set contains arguments of entity type, like Time and Place along with more complex argument types, like After Effects and Reason. These arguments with non-entity type attributes have comparatively much fewer instances in the data set than the entity type arguments. Not only that, these arguments also constitute a much more complex lexical and syntactical structure and are hard to capture using the existing architectures. In our work, we propose a novel event feature that helps capture rare complex argument instances more accurately than the existing models on this task. We construct a causal knowledge structure with the aid of external knowledge bases and for a given event, we extract its corresponding causal attribute from the structure.

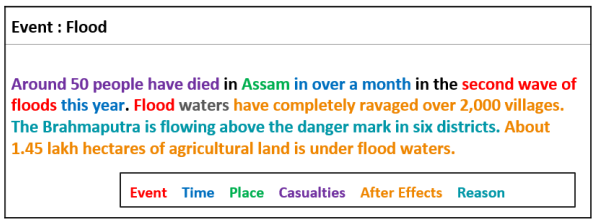


Figure 1: Sample document with annotated events and arguments. The global event category of the document is *Flood* and the words in red indicate the event triggers.

Context has always been of paramount importance for the task of event argument extraction. Existing literature outlines different paradigms of modeling the contextual scope for this task. The

Proceedings of the 17th International Conference on Natural Language Processing, pages 287–296
Patna, India, December 18 - 21, 2020.

traditional sentence level event argument extraction tasks performed on popular datasets like ACE 2005 and TAC-KBP 2017 largely restrict their contextual scope to within sentence boundaries (Chen et al., 2015; Subburathinam et al., 2019). However, some researchers identified the need to mine global contextual features to enhance the argument extraction capabilities and modelled the task at a multi-sentence or paragraph level (Yang and Mitchell, 2016; Duan et al., 2017; Zhao et al., 2018), and at a document level (Yang et al., 2018; Zheng et al., 2019). In our work, we also identify the need to explore cross-sentence contextual scope to aid accurate extraction of events and argument spans. For example as illustrated in Figure 1, the sentence *"The Brahmaputra is flowing above the danger mark in six districts."* could refer to a *Reason* or an *After-Effect* argument in different contexts. But on reading the document, one can understand that the document talks about a *Flood* event, and that the sentence can be tagged as *Reason* argument with considerable confidence. Event arguments do not necessarily co-occur in the sentence containing the event trigger. This behaviour is even more frequently observed in longer documents like news articles. With no thematic signals obtained from event trigger words, it becomes difficult to identify the correct argument labels and their spans in such cases. In our work, we look beyond sentences for stronger contextual clues to better capture the events and their arguments from a document.

In summary, the **contributions** of our work are four fold :

1. We propose a novel event causal feature for improved extraction of complex causal arguments.

2. Through our work, we emphasize on the effectiveness of global vs. local scope of contextual information in this task.

3. We compare different approaches of modeling global information in this task and evaluate the effectiveness of each.

4. We provide a novel end-to-end system for event argument extraction which beats the state-of-the art model's performance.

1.1 Terminology

We introduce certain terminologies to facilitate better understanding of our work .

- Event : An **event** is any real life happening or occurrence which may be denoted by a word or a phrase.

- Event Trigger : The particular word or phrase that evokes a particular event type is known as an **event trigger**.

- Argument : The words or phrases which provide supporting information about the event are known as **arguments**.

- Causal Argument : The causal arguments refer to the **Reason** & **After Effect** arguments of an event. The *Reason* argument holds information regarding why a particular event happened whereas *After Effects* argument details information about the after-math of an event. It is of importance to note here that the deaths and injuries that occur because of an event are noted under the *Casualties* argument whereas all other effects of an event are covered under the *After Effects* argument .

- Entity : Words or phrases which specifically refer to terms that represent real-world objects like people, places, organizations are known as **entities**.

2 Related Work

Event Argument Extraction is a well researched domain, primarily in English (Chen et al., 2015), (Nguyen et al., 2016) and Chinese (Lin et al., 2018), (Yang et al., 2018). While there have been efforts to extend this task to a limited number of languages like Spanish and Arabic (Akbik et al., 2016), (Subburathinam et al., 2019), the task has been hardly explored in Indian languages. In this paper, we make an effort to explore five out of the twenty two commonly spoken languages of India. The existing literature for Indian languages for this task are minimal and have either extracted arguments from unstructured text irrespective of their event links (Ahmad et al., 2019), or have jointly extracted event and argument mentions with rule based approaches (Patel et al., 2019). Each of these works are modelled to focus at an intra-sentence local scope and do not capture inter sentence dependencies. While the sentence level extraction mechanisms outlined by prior works present a promising preface for this task in low-resource languages, the results are indicative of a huge scope of improvement - especially for arguments like *Reason*

which are scant on annotations. In our work, we try to address this gap and extract sentence level events and arguments by infusing document level context. Infusing cross-sentence or global context to extract sentence level information has already shown great improvements in many related works. The work by (Zhang et al., 2020) studies the event-argument extraction task by infusing contextual information from the five sentences surrounding an event trigger. Their work highlights the fact that learnt architectures usually capture arguments from sentences containing event triggers better than from non-event-trigger sentences. Addressing this gap, we adopt a trigger-less event detection approach in our work, similar to (Zheng et al., 2019) and use document-level event information to mitigate the dependence on local event clues obtained from event triggers. The method adopted by (Wadden et al., 2019) uses a graphical span enumeration approach with a document input to model cross-task and inter-sentence dependencies. We model our work to include inter-sentential context to extract event-argument spans from input documents with greater accuracy.

3 Dataset

We have curated an event argument annotated dataset for English and four Indian languages, namely Bengali, Hindi, Marathi and Tamil as part of a collaborative effort. The dataset comprises of documents with sentence-level annotations of events and argument mentions. The data set caters specifically to the disaster domain and covers 32 event types at a fine grain level and 12 event types at a coarse level. The dataset contains annotations for 14 argument types, but in this work we focus on 6 main argument types which are, *Time, Place, Casualties, After-Effect, Reason and Participant.* Almost all possible man-made and natural disaster event types have been covered in this corpus. Typically used datasets in this task like ACE 2005 and TAC KBP cater to generic events with only entity type arguments. This corpus contains arguments of both entity type and non entity type with widely varying argument boundaries. Non-entity type arguments constitute of complex argument types like *Reason* and *After-Effects* which may constitute one/many entities within it's span. This makes the task even more challenging and contributes to it's uniqueness. The raw data for the corpus was collected from the FIRE document repository as well as by crawling popular news websites for the respective languages. The dataset was annotated by eight linguistic experts over a span of two years. The multi-rater Kappa agreement ratio of the annotators has been evaluated as 0.85 approximately. The data distribution is reported in Table 1

Language	Train	Valid	Test
English	456	56	131
Bengali	699	100	199
Hindi	678	150	194
Marathi	815	117	233
Tamil	1085	155	311

Table 1: The number of documents for each language in their Train, Test and Validation splits.

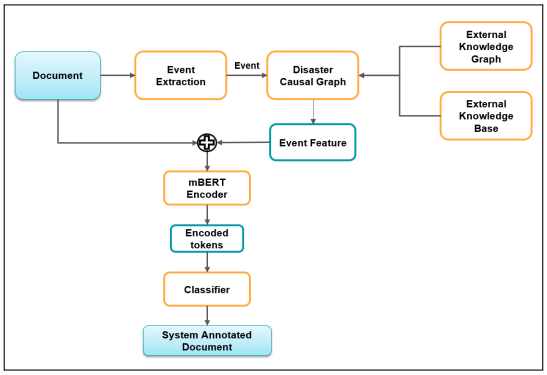


Figure 2: Overview of the general architecture.

4 Approach

In this section we elaborate on the approaches we have taken to accomplish the task of extracting the event labels and it's corresponding arguments from a document. This is essentially a two-fold task: i) Document Level Event Extraction, and ii) Argument Extraction. The generic overview of our approach is illustrated in Figure 2.

4.1 Problem Formulation

In our work, we treat the task of event-argument extraction as a sequence labeling task. Given N training instances $D = (x_n, y_n)_{n=1}^{N}$, where x is a sequence of tokens from an input document and y is the corresponding event-argument annotation sequence for x, our aim is to build a neural model which maximises the conditional probability $p_\theta = (y|x)$, parameterized by model parameters θ. To capture events and arguments better, we propose modeling context at a global scope for this task. Infusion of global context in our work is done

Language	Micro F1-Score (%)
English	97.71
Bengali	68.34
Hindi	87.63
Marathi	78.54
Tamil	84.89

Table 2: Document Level Event Extraction Results

in two ways. First, instead of processing each sentence as a training instance, we process an entire document with sentence demarcations. Second, we infuse the document level event information into the model to aid more accurate identification of complex argument types. In the sections to follow, we have described the different approaches we have adopted to process event information into the model. We have conducted our experiments on five Indian languages.

4.2 Document Level Event Extraction

Traditional approaches for event extraction involve extracting event triggers at a sentence level. We adopt a no-trigger approach to label the documents with their corresponding events but instead of adopting a distant supervision method like (Zheng et al., 2019), we adopt a supervised approach with the aid of pre-trained language models in this task. Our dataset comprises of news articles, and most often, it has been observed that the central theme of such documents are divulged in the title of the document itself. Moreover, we find that only 6% of the total corpus contains documents with multi-event instances. Hence, we formulate this task as a multi-class sentence classification task where each document corresponds to one event category. We consider the event labels of the first event trigger instance in each document as the document's manually assigned event label. We fine-tune a pre-trained multilingual BERT (mBERT) encoder with a linear classification head on top to classify each document title instance to an event category. Since the titles of the documents are not explicitly defined, we take the first two sentences as the title of the news article and use it as our training and testing instances. We use the the pooled output of the encoder as input to the linear classification head in the model. We find this approach to be highly effective with a very high percentage of documents being assigned their correct event labels as shown in Table 2.

4.3 Argument Extraction

The main focus of this sub-task is to extract the sentence level arguments given the document-level event information. To aid the extraction of causal arguments, we introduce a causal feature along with the input document to the argument extraction module. In the following sections, we discuss how we build and use the causal feature for event argument extraction.

4.3.1 Event Causal Feature

Unlike the generic entity type arguments like *Time* and *Place*, causal type arguments follow complex patterns, ranging over variable lengths and show a strong reliance on the event category. This is a natural observation - the cause and effects of an Earthquake will not be similar to that of a flood or a terrorist attack. To make the task even more challenging - these argument types, especially the *Reason* argument, have very few training instances in the corpus. We define a novel event causal feature in our approach to aid the extraction of causal arguments.

Causal Graph Construction We manually construct a disaster event graph ontology based on the structures defined in the annotation guidelines of the corpus as well as background knowledge of linguistic experts. Given a graph $G = (V, E)$, the vertices V correspond to events and their causal argument roles (Reason & After-Effects) and the edges E show the relationships among the various events and their arguments. We manually define the event-event relationships in the graph to aid knowledge transfer across events, such that empty nodes in the graph corresponding to argument roles, can inherit knowledge for that specific argument role.

Node population The event nodes contain the event-types. The children nodes correspond to the causal argument roles. These nodes are populated using words and phrases from external knowledge bases like ConceptNet (Speer et al., 2017) and Wikipedia. For ConceptNet, we have identified a few edge-relations which correspond to our causal argument labels and nodes that correspond to our event types. It is worth mentioning here, that not all the event types could be accessed in the external resources. To mitigate that, we used a list of manually curated synonyms of the event types or merged some related event types (like *Transport*

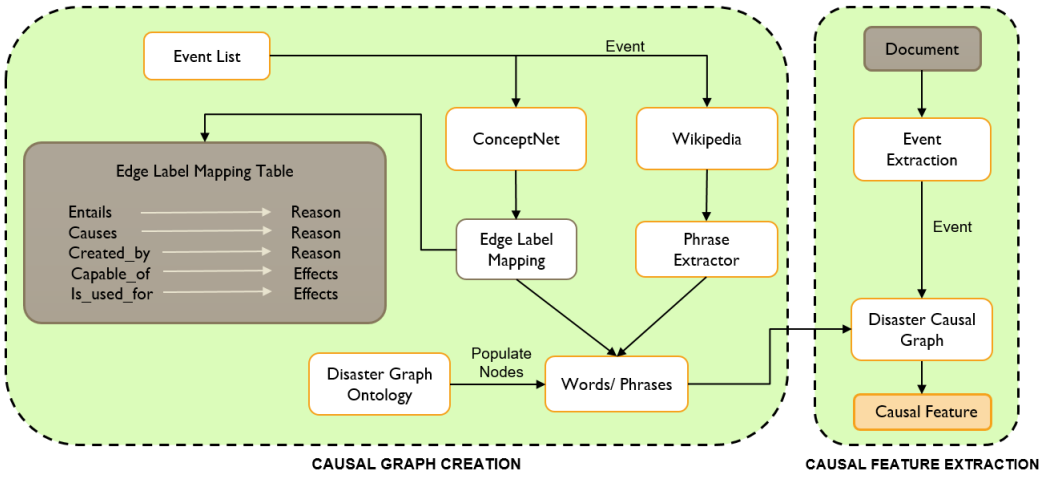


Figure 3: Detailed overview of the causal feature construction process. Since our data is of Disaster domain, we have labeled the graph in the diagram as Disaster Causal Graph.

Hazards and Vehicular Collision) to find relevant resources in the external knowledge bases. For each event type, we query the nodes associated with the identified edges, extract the words and phrases present in the nodes and use that to populate our corresponding causal graph nodes. We also crawl the cause and effects section of relevant Wikipedia pages corresponding to the event types in our corpus. We use an unsupervised language-agnostic phrase extractor (Campos et al., 2020) to extract phrases from the crawled Wikipedia sections and populate the corresponding causal graph nodes with the extracted phrases. Each argument node in the graph maintains a minimum of 3 and a maximum of 10 words or phrases. In the event that a certain argument node in the causal graph has less than 3 words or phrases, we exploit the defined event-event relations in the causal graph and inherit values from related event-argument nodes. The entire process is illustrated in Figure 3.

In an additional final step, we use a sentence template to string together the words and phrases of the argument nodes of a particular event. The constructed event-causal sentence for each event will henceforth be referred to as the event causal feature in the paper.

4.3.2 Argument Span Identification & Classification

Following the pipeline illustrated in Figure 2, for each document, we extract it's event category and use the extracted event to obtain the corresponding event causal feature. Each input instance to the argument extraction module is augmented with the event causal feature at both ends. We train all the 12 encoder layers of the pre-trained bert-base-multilingual-cased model along with an additional linear classification head on top. The linear layer classifies the hidden states output corresponding to each token to one of the six argument types (*Time, Place, Casualties, After-Effects, Reason, Participant*) or to *Others*.

In the sections to follow, we will refer to our approach defined in this section (Section 4) as **Event Causality Augmentation** or **ECA**.

5 Results

5.1 Experimental Settings

We use Huggingface's bert-base-multilingual-cased model pre-trained on 104 languages[1]. Adam(Kingma and Ba, 2014) was used to optimize the parameters. The model was trained with a mini-batch size of 4 on a single Tesla k40-C machine and a maximum sequence length of 512. The models were trained for 20 epochs or until no significant changes were observed in the validation loss, whichever was earlier.

5.2 Baselines

We compare our work with the following enlisted approaches:

1. Patel-Emb: (Patel et al., 2019) are considered the state-of-the-art results on this dataset. The works of (Patel et al., 2019) employ a set of rules to create a rule vector for each token

[1]https://huggingface.co/bert-base-multilingual-cased

Models	Time	Place	Casualties	After-Effects	Reason	Participant	Avg.
Patel-Emb	83.05	88.41	90.09	55.90	81.56	76.40	79.23
Patel-Parallel	84.65	87.48	89.09	56.48	86.31	75.42	79.90
Patel-KD	81.74	87.19	89.83	47.79	87.35	75.45	78.23
mBERT	91.50	92.80	93.50	55.30	85.90	84.00	83.80
JETAE	90.37	91.84	92.87	47.42	77.07	83.41	80.49
EA	90.90	92.90	93.60	54.90	83.60	86.40	83.71
ECA (Our model)	**92.65**	**93.41**	**94.22**	**58.39**	**90.76**	**87.29**	**86.12**

Table 3: Argument-Wise comparison of our approach (ECA) with other baseline approaches on the English Corpus using F1-scores. The bold text refers to the best result obtained for each argument as well as overall (Avg.).

which embeds event class information. They use different approaches to infuse the rule vector into the model. In this approach, the rule vector was appended with each sentence token representations and fed to a single layer Bi-LSTM model.

2. Patel-Parallel : In this approach of (Patel et al., 2019), the rule vector and word embeddings were fed to two parallel single layer Bi-LSTM models. Their learnt hidden layer representations were concatenated to learn a joint representation of words and rules.

3. Patel-KD: In this approach of (Patel et al., 2019), the knowledge of rules is distilled to the Bi-LSTM network by using the rule vector to bias the weights of the neural network.

4. mBERT: We adopt the BERT-NER from (Devlin et al., 2019) and fine-tune the mBERT Token Classifier to extract event-argument spans and compare it with our feature based mBERT argument extractor.

5. JETAE: We adapt the mBERT Token Classifier for **J**oint **E**vent **T**rigger and **A**rgument **E**xtraction task as defined in (Patel et al., 2019) and report it's argument extraction capabilities as compared to our approach.

6. EA (**E**vent **A**ugmentation): Instead of augmenting the event causal feature, we augment the event type at both ends of the document and compare it's results against our's.

We compare the overall performance in Table 3 for English. For the Indian languages, we compare the F1-scores we have obtained using our approach for each argument type against the corresponding F1 score obtained from (Patel et al., 2019) in Table 5. Since none of the three approaches defined in (Patel et al., 2019) were reported as the best performing method for the task, for a given natural language, we have considered the highest F1-score obtained for each argument type among the three approaches as the state-of-the-art result for the same. We report our results using precision, recall and f1-scores.

Arg. Type	ECA-gold	ECA-pred
Time	91.84	92.19
Place	71.81	70.82
Casualties	82.96	82.79
After-Effects	53.31	53.13
Reason	42.71	39.88
Participants	55.11	54.00

Table 4: Results to highlight error propagation for the Bengali corpus. We report the F1-scores for each argument type with manually annotated event labels (ECA-gold) and predicted event labels (ECA-pred).

6 Analysis

We perform a thorough analysis of our approaches and enlist their merits and demerits in this section. We compare our models to the state-of-the-art models established on this dataset. As reported in Table 3, we establish new state-of-the-art results on this dataset. We present a comparative analysis of the approaches explored in our work and observe that our approach **E**vent **C**ausality **A**ugmentation (**ECA**) reports the best performance (Table 3). In the table, we can observe a huge performance boost with the use of fine-tuned contextualised word representations of mBERT. The intended performance boost in the extraction of causal arguments is also observed on introduction of the causal feature. We report the argument extraction performance of ECA with both manually annotated event labels (ECA-gold) and event labels predicted by our event extraction module (ECA-pred) in Table 4. We specif-

Lang.	Arg. Type	SOTA	ECA	Δ%
Bengali	Time	88.59	92.19	+4.06
	Place	68.20	70.82	+3.84
	Casualties	78.95	82.79	+4.86
	After Effects	41.54	53.13	+27.90
	Reason	17.91	39.88	+122.67
	Participant	49.93	54.00	+8.15
Hindi	Time	68.57	72.54	+5.79
	Place	63.79	73.06	+14.53
	Casualties	68.21	71.93	+5.45
	After Effects	44.67	45.43	+1.70
	Reason	19.25	12.42	-35.48
	Participant	43.18	56.24	+30.25
Marathi	Time	75.59	80.97	+7.12
	Place	69.44	74.74	+7.63
	Casualties	75.69	79.48	+5.01
	After Effects	47.29	59.18	+25.14
	Reason	35.93	35.57	-1.00
	Participant	59.35	64.55	+8.76
Tamil	Time	81.03	89.09	+9.95
	Place	75.86	83.84	+10.52
	Casualties	80.20	87.21	+8.74
	After Effects	72.15	83.82	+16.17
	Reason	42.72	67.94	+59.04
	Participant	65.71	74.61	+13.54

Table 5: Comparison of Argument Extraction Performance of our model (ECA) with the state-of-the-art (SOTA) results ((Patel et al., 2019) for each language. We show the % increase (+) or decrease (-) in F-Scores with respect to the SOTA results.

Ground Truth: सड़क निर्माण के दौरान <Time> पहाड़ी के गिरने से मलबे में दबकर दो मजदूरों <After-Effects> की मौत हो गई।

Translation: While road construction<Time>, due to landslide, two workers<After-Effects> were crushed under the debris and killed.

Predicted: सड़क निर्माण के दौरान पहाड़ी के गिरने से <Reason> मलबे में दबकर दो मजदूरों की मौत <Casualties> हो गई।

Translation: While road construction, due to landslide<Reason>, two workers were crushed<Casualties> under the debris and killed.

Figure 4: Manually annotated and Inferred example from the Hindi dataset highlighting the systemic confusion between *event triggers* and *reason* arguments. It also highlights the annotation errors existing in the corpus, which is discussed in Section 6.4.

ically show these results on the Bengali dataset as it had reported the weakest event extraction results among the five languages in Table 2. In table 2 we observe that the F1-scores for Bengali and Marathi are considerably low compared to the other languages. The drop in performance can be attributed to the fact that these two languages contain the maximum number of multi-event documents while our approach is based on the assumption that each document in the corpus is a single event document. Even in multi-event documents, we had observed that the dominant event was mostly present in the title of the document. The Bengali corpus consisted of a few exceptional multi-event news articles which contained news snippets about completely unrelated topics. In such cases the event extraction algorithm performs poorly, thus reporting a significant drop in performance on the Bengali dataset for the document level event extraction task.

We can observe the error from the event extraction module propagating to our argument extraction method in ECA-pred with the effects being visible mostly in the causal arguments in Table 4. We leave the task of a joint event label identification and argument extraction as a future task, which should be able to mitigate this problem. In both Table 3 and Table 5, we can see the desired improvement for the causal arguments. For almost all arguments we notice a major improvement in the performance in Table 5. In Hindi & Marathi though, we observe a dip in the performance for the *Reason* argument. We find that the dip can be attributed to two factors: i) the model confuses event triggers in certain cases as the *Reason* argument - events can be the reason for other events in the document as well; ii) there are *Reason* argument instances which are not annotated but gets detected by the model. We have illustrated (i) through an example in Figure 4. In the example, it can be observed that the model confuses the event trigger *due to landslide* as the *Reason* for the death of two workers. Although, it is correct in it's essence, as per the task, the phrase *due to landslide* is an event trigger and not part of a *Reason* argument.

Ground Truth

ডায়মণ্ড হারবারের এস ডি পি ও দীপশঙ্কর রুদ্র জানান আচমকা বাসটির সামনের চাকা ফেটে যাওয়ায় <Reason> রাস্তা থেকে ছিটকে সেটি পাশের একটি **খালে কাত হয়ে পড়ে যায়।** **<event trigger>**

Translation: *Diamond Harbour's SDPO Dipshankar Rudra has informed that due to a sudden burst of the bus's front tires<Reason>, it skid from the road, **overturned and fell into a ditch.<event trigger>***

<Document Event Type : Transport Hazards >

Model Type	Predicted Instance
mBERT (Without Event)	No Prediction
EA	ডায়মণ্ড হারবারের এস ডি পি ও দীপশঙ্কর রুদ্র জানান আচমকা বাসটির<Participant> সামনের চাকা ফেটে<Reason> যাওয়ায় রাস্তা থেকে ছিটকে সেটি পাশের একটি খালে কাত হয়ে পড়ে যায়। **Translation**: *Diamond Harbour's SDPO Dipshankar Rudra has informed that due to a sudden burst of the<Reason> bus's<Participant> front tires<Reason>, it skid from the road, overturned and fell into a ditch.*
JETAE	No Prediction
ECA	ডায়মণ্ড হারবারের এস ডি পি ও দীপশঙ্কর রুদ্র জানান আচমকা বাসটির সামনের চাকা ফেটে যাওয়ায় রাস্তা থেকে<Reason> ছিটকে সেটি পাশের একটি খালে কাত হয়ে পড়ে যায়। **Translation**: *Diamond Harbour's SDPO Dipshankar Rudra has informed that due to a sudden burst of the bus's front tires<Reason>, it skid from the road <Reason> , it skid from the road, overturned and fell into a ditch.*

(a) Example 1

Ground Truth

কৃষ্ণনগর স্কুলের মাঠে <Place> গাড়ি চালানো শেখার সময় <Time> চালক নিয়ন্ত্রণ হারিয়ে ফেলায় <Reason> গাড়িটির চাকায় পিষ্ট হয়ে তৃতীয় শ্রেণীর এক ছাত্রের মৃত্যু <Casualties> হয়েছে.

Translation : *While taking driving lessons<Time> in Krishna-nagar School grounds<Place>, the driver lost control<Reason> and a 3rd grade student died<Casualties> on getting crushed under the wheels of the car.*

<Document Event Type : Transport Hazards, sentence has no event trigger>

mBERT (Without event)	কৃষ্ণনগর স্কুলের মাঠে <Place> গাড়ি চালানো শেখার সময় চালক <Reason> নিয়ন্ত্রণ হারিয়ে ফেলায় <After Effects> গাড়িটির চাকায় পিষ্ট হয়ে তৃতীয় শ্রেণীর এক ছাত্রের মৃত্যু <Casualties> হয়েছে। **Translation:** *While taking driving lessons in Krishnanagar School grounds<Place>, the driver<Reason> lost<After Effects> control and a 3rd grade student died<Casualties> on getting crushed under the wheels of the car.*
EA	কৃষ্ণনগর স্কুলের মাঠে <Place> গাড়ি চালানো শেখার সময় চালক নিয়ন্ত্রণ হারিয়ে<After Effects> ফেলায় গাড়িটির চাকায় পিষ্ট হয়ে তৃতীয় শ্রেণীর এক ছাত্রের মৃত্যু <Casualties> হয়েছে। **Translation:** *While taking driving lessons in Krishnanagar School grounds<Place>, the driver lost control<After Effects> and a 3rd grade student died<Casualties> on getting crushed under the wheels of the car.*
ECA	কৃষ্ণনগর স্কুলের মাঠে <Place> গাড়ি চালানো শেখার সময় চালক নিয়ন্ত্রণ <Reason> হারিয়ে ফেলায় গাড়িটির<Participant> চাকায় পিষ্ট হয়ে তৃতীয় শ্রেণীর এক ছাত্রের মৃত্যু <Casualties> হয়েছে। **Translation:** *While taking driving lessons in Krishnanagar School grounds<Place>, the driver lost control<Reason> and a 3rd grade student died<Casualties> on getting crushed under the wheels of the car<Participant>*

(b) Example 2

Figure 5: Various inferred examples collected for a thorough qualitative analysis. The different approaches compared are our model (ECA), Event Augmentation (EA), without event (mBERT), and Joint Event Trigger and Argument Extraction model (JETAE).

6.1 Importance of event causal information

To investigate the importance of event information in our model for the task of argument extraction, we present our findings in Table 6, where we compare mBERT based argument extraction models without event information (without-event), with event type information (EA) and finally with event causal feature (ECA - our model). We can see a significant improvement in performance with event information, especially with the causal feature. This rise can mainly be attributed to the significant improvement in causal argument extraction capabilities which rely heavily on event information. We also qualitatively analyse our findings in the first example cited in Figure 5. As can be observed, without event information, it is unable to detect the presence of a *Reason* Argument. With event type information, the model is able to detect the argument's presence, but we observe the model suffering from ambiguity over argument types. This ambiguity often rises with complex argument types being nested among each other, as can be observed in this example. *Bus* is a Participant in this event but in this example, it is part of longer *Reason* argument span. With addition of the causal feature, we find causal contexts getting learnt better and as observed in this example, causal argument spans get captured efficiently.

6.2 Importance of document level event information

Extracting document level event information has multiple benefits compared to event trigger extraction: i) It helps capture the overall thematic preface of the document, ii) By not extracting event triggers, we avoid the problem of ambiguous event triggers for multiple event types. For example, the word *explosion* can be an event trigger for event types *Terrorist Attacks, Volcano*, as well as *Industrial Accidents*.; iii) It saves annotation labour and cost. Labeling document-level events is much simpler a task than annotating event triggers in each sentence of the document. By taking into account the thematic preface of the document, we will now observe it's effects on the task of event argument extraction using the same example as we used above (first example of Figure 5). In JETAE, we often observe the model exhibiting bias towards sentences with event triggers when it comes to the task of argument extraction. In this example, since the

model was unable to detect the event trigger in the sentence, it is unable to detect it's corresponding argument as well. Because of the use of document level event context, we can observe that both EA and our model, ECA are able to detect the *Reason* argument span.

6.3 Sentence vs. Paragraph vs. Document

We study the importance of document context in the task of extracting arguments of an event. To investigate the importance of the contextual scope, we have experimented in three different settings by formatting the input instance to either include a single sentence, a paragraph or the entire document. Since paragraph boundaries were not mentioned in the dataset, we have assumed the paragraphs in our work to constitute four sentences. This decision was taken after finding the average length of the documents to be 15 sentences. We present our findings in Table 7 where we can observe a consistent increase in the F1-scores as we widen our contextual scope. The scores in each setting were evaluated by averaging the individual argument scores. While the performances at sentence and paragraph level are comparable, it is in document level context that a significant increase is observed. This increase can be mainly attributed to the effective extraction of the scattered arguments with no localised event clues to aid in their extraction process.

6.4 Case Study

In this section, we analyse a few inferred examples across different approaches adopted in our task. Apart from the points already discussed, we observe a few more challenges and their solutions along with their merits and demerits through our approaches. If we analyse the second example in Figure 5, we find a classic confusion between the *Reason & After Effect* arguments. Reason and after-effects are often interchangeable concepts depending on the contextual premise. As can be observed, addition of causal feature helps resolve this ambiguity. We also observe that the model efficiently captures arguments which were otherwise missed by human annotators like the Participant argument *car* in this example. This is primarily because of the document context and other similar Participant instances being annotated in likewise document context that the model is able to capture the missed annotations in the document as well. We also observe, across examples that the argument boundary

	P	R	F
Without-event	66.32	62.57	63.45
EA	67.21	63.45	64.14
ECA	67.91	65.69	66.20

Table 6: Comparison of different approaches on the Bengali Dataset to establish the importance of event information and event causal feature in our task.

Model	Sentence	Paragraph	Document
EA	59.59	62.59	64.14
ECA	61.20	62.42	66.20

Table 7: Comparison of argument-extraction F1-scores with respect to the contextual scope, for the Bengali dataset of our corpus.

mismatch persists in our approaches as well. The argument window may not be precise and may differ by a few tokens on either ends of the span.
Another observation that we make is more in lieu of the annotation errors in the corpus which the model has been observed to overcome in many situations. A confusion among annotators was observed about the semantic scope of *Casualties* and *After Effect* arguments as observed through the example in Figure 4. However, the model in most cases extracts the argument with the correct label (as also illustrated in Figure 4) and thus, exhibits the model's robustness against noise.

7 Conclusion

In our work we have shown a pipeline approach to mine document level event labels and their arguments from each sentence in a document. We have reported a comparison study across different approaches based on multiple parameters, like input context and event information. In future, we want to extend this task to handle more languages in a multilingual setting. Our approaches have shown a huge improvement over the state-of-the art methods. In conclusion, through our work, we emphasize on the importance of both contextual and event information and report our thorough investigations on the same.

Acknowledgments

The work done in this paper is an outcome of the project titled "A Platform for Cross-lingual and Multi-lingual Event Monitoring in Indian Languages", supported by IMPRINT-1, MHRD, Govt. of India, and MeiTY, Govt. of India.

References

Zishan Ahmad, Deeksha Varshney, Asif Ekbal, and Pushpak Bhattacharyya. 2019. Multi-linguality helps: Event-argument extraction for disaster domain in cross-lingual and multi-lingual setting.

Alan Akbik, Laura Chiticariu, Marina Danilevsky, Yonas Kbrom, Yunyao Li, and Huaiyu Zhu. 2016. Multilingual information extraction with polyglotie. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 268–272.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, A. Jorge, C. Nunes, and Adam Jatowt. 2020. Yake! keyword extraction from single documents using multiple local features. *Inf. Sci.*, 509:257–289.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. 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 167–176.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Shaoyang Duan, Ruifang He, and Wenli Zhao. 2017. Exploiting document level information to improve event detection via recurrent neural networks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 352–361, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2018. Nugget proposal networks for chinese event detection. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1565–1574.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309, San Diego, California. Association for Computational Linguistics.

Hrishikesh Patel, Nandan Rathod, Ayush Maheshwari, Ritesh Kumar, Ganesh Ramakrishnan, and Pushpak Bhattacharyya. 2019. Tale of tails using rule augmented sequence labeling for event extraction. *arXiv preprint arXiv:1908.07018*.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: an open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 4444–4451.

Ananya Subburathinam, Di Lu, Heng Ji, Jonathan May, Shih-Fu Chang, Avirup Sil, and Clare Voss. 2019. Cross-lingual structure transfer for relation and event extraction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 313–325.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5788–5793.

Bishan Yang and Tom Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 289–299.

Hang Yang, Yubo Chen, Kang Liu, Yang Xiao, and Jun Zhao. 2018. DCFEE: A document-level Chinese financial event extraction system based on automatically labeled training data. In *Proceedings of ACL 2018, System Demonstrations*, pages 50–55, Melbourne, Australia. Association for Computational Linguistics.

Zhisong Zhang, Xiang Kong, Zhengzhong Liu, Xuezhe Ma, and Eduard Hovy. 2020. A two-step approach for implicit event argument detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7479–7485, Online. Association for Computational Linguistics.

Yue Zhao, Xiaolong Jin, Yuanzhuo Wang, and Xueqi Cheng. 2018. Document embedding enhanced event detection with hierarchical and supervised attention. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 414–419, Melbourne, Australia. Association for Computational Linguistics.

Shun Zheng, Wei Cao, Wei Xu, and Jiang Bian. 2019. Doc2edag: An end-to-end document-level framework for chinese financial event extraction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 337–346.

Claim Extraction from Text using Transfer Learning

Acharya Ashish Prabhakar[1], Salar Mohtaj[1,2], and Sebastian Möller[1,2]

[1]Quality and Usability Lab, Technische Universität Berlin, Berlin, Germany
[2]DFKI Projektbüro Berlin, Berlin, Germany
a.prabhakar@campus.tu-berlin.de
{salar.mohtaj|sebastian.moeller} @ tu-berlin.de

Abstract

Building an end to end fake news detection system consists of detecting claims in text and later verifying them for their authenticity. Although most of the recent works have focused on political claims, fake news can also be propagated in the form of religious intolerance, conspiracy theories etc. Since there is a lack of training data specific to all these scenarios, we compiled a homogeneous and balanced dataset by combining some of the currently available data. Moreover, it is shown in the paper that how recent advancements in transfer learning can be leveraged to detect claims, in general. The obtained result shows that the recently developed transformers can transfer the tendency of research from claim detection to the problem of check worthiness of claims in domains of interest.

1 Introduction

The advent of social media and mobile-based messaging applications has led to a rapid spread of fake news and misinformation, which are having serious consequences in real-world scenarios. Manual fact-checkers are often overwhelmed by the number of sources that need to be verified. Facebook has released some statistics in its regular enforcement reports, as follow:

- In 2016, known fake news content was getting around 200 million engagements on Facebook each month
- In Q1 of 2018, Facebook removed 837 million pieces of spam
- In Q1 2018, Facebook also removed 583 million fake accounts

The automated fact-checking process can help to mitigate this problem. The very first step of a fact-checking system is to identify claims from the text, which can then be verified by querying a knowledge base or by converting it into a question and looking for suitable answers using a crawler (Vlachos and Riedel, 2014). Since identifying claims is the very first step of a fact-checking pipeline, the performance on this task has a major impact on the result of an end to end fact-checking system. As the performance of the state-of-the-art deep learning models are highly dependent to the availability of labelled data, in this paper, we introduce a novel dataset to assist in training models for the task of claim detection. Moreover, we investigate the performance of some of the recent transformers based language models (e.g., BERT) on the proposed data.

The paper is organized as follow. Some of the recently proposed models and datasets for the task of claim detection are presented in Section 2. Section 3 contains a brief introduction to transfer learning and the state-of-the-art models. The proposed approaches for compiling the dataset and detecting claims, and also the baseline model are described in details in Section 4.

2 Related Work

Since fact-checking and fake news detection is time-consuming and effort-full tasks, in many systems, claim detection and argument mining are considered as the preliminary modules that provide input for fact-checking module.

ClaimBuster, which is introduced in (Hassan et al., 2017), tries to rank political claims in debates based on their check-worthiness using supervised models. To this end, a labelled dataset of spoken sentences by presidential candidates is constructed. Each sentence in the dataset is given one of the three possible labels; it is not a factual claim; it is an unimportant factual claim; it is an important factual claim. Among various classification

Proceedings of the 17th International Conference on Natural Language Processing, pages 297–302
Patna, India, December 18 - 21, 2020.

Method	Source of training data	Model used	Precision	Recall	F1
ClaimBuster	Hand annotated US presidential debates	SVM	79%	74%	76%
ClaimRank	Popular fact checking organizations	FNN	93%	65%	77%
Full Fact	Crowd sourced annotations	LogReg	88%	80%	83%
Logically	Annotated news articles	Google USE	90%	89%	89%

Table 1: Comparison of the scores of previous works

methods which have been trained on the proposed dataset, support vector machine outperforms the other methods in the accuracy of finding important claims. Its *ClaimSpotter* component performs the task of claim detection with a precision of 79% and a recall of 74%.

Konstantinovskiy et al. (2018) focused on monitoring news sources and identifying "if a particular sentence constitutes a claim that could be fact-checked?". Since the definition of claim and also its importance are subjective tasks and rely on many factors (e.g. personal background), one of the most important aspects of their work is proposing an annotation guideline in which 'claim' and 'important claim' are defined to annotators. Their final results show that logistic regression classifier gives the highest overall F1 score, comparing to the other supervised models in a different setting.

Another recent approach to claim detection is *ClaimRank* (Jaradat et al., 2018). The authors claim that although the system originally trained on political debates, it works for any text, e.g. interviews and regular news articles. They compiled a dataset by taking the outputs of fact-checking of a political debate, published by nine reputable organizations simultaneously. Models were created to predict if the claim would be highlighted by at least one or by a specific organization. The modelling is done with a large variety of features from both the individual sentence and the wider context around it. To classify claims and rank them on their check worthiness, a two hidden layered neural network is trained. Adler and Boscaini-Gilroy (2019) used Google's universal sentence encoder to obtain the sentence embedding and passed it through a logistic regression layer to get the final classification. Their model has trained on a dataset based on news articles created by them.

A detailed comparison of the previously proposed methods is presented in Table 1. In addition to develop a claim detection based on transfer learning approach, in this paper we compile a new balanced dataset for the claim detection task, containing sentences from different domain and contexts.

3 Transfer Learning

Neural networks need large scale datasets to be trained efficiently. They are difficult to apply where the available data is sparse. The Imagenet moment (Deng et al., 2009) where fine-tuning, a model trained on a large dataset could be applied in various applications with limited availability of data, created a breakthrough in computer vision.

Language modelling was seen as the most appropriate task to model this achievement by Imagenet in Natural Language Processing (NLP). When trained for language modelling, neural networks capture the basic structure and understanding of language and can thus be fine-tuned on downstream tasks with limited data. Many attempts have been made in creating a sophisticated model trained on a large dataset that can be fine-tuned easily on a downstream task. *ULMFit* (Howard and Ruder, 2018) was one of the earliest models to achieve this through an *AWD LSTM* (Merity et al., 2018). *ELMO* (Peters et al., 2018) used a bi-directional *LSTM* architecture with character level encoding of features to avoid out of vocabulary errors. *BERT* (Devlin et al., 2019) used the recently developed transformer architecture to perform language modelling.

Transformers have the advantage of being able to train faster due to possibility of parallelization. *DistilBert* model (SANH et al.) was created by a method called model distillation, and it is 40% smaller, and 60% faster than *BERT* and it retains 97% of its performance, making it more deployment friendly. In our experiments, *BERT* and *DistilBert* have been used to extract claims from text.

4 Proposed Approach

In this section we present the applied approaches for compiling the dataset and detecting claims. Moreover, the baseline model is also described in this section.

Description		No.
Document statistics	Total number of samples	395,057
	Total number of claim samples	197,528
	Total number of non-claim samples	197,529
Word/Char statistics	Average number of words per sample	83.61
	Average character per sample	408.94
	Average number of stop words per sample	39.18

Table 2: Statistics of our dataset.

4.1 Dataset

To train a neural network for detecting claims, we need a dataset with claim and non-claims classes. There are several datasets available for the task of claim detection. *FEVER* (Thorne et al., 2018) is the largest available dataset for this task. Their annotators performed several types of mutations of *Wikipedia* articles summary section to create claims. Wang (2017) collected claims from well-known fact-checking organizations. However, there are no significant datasets available for the negative class (i.e. non-claims). Since several machine learning methods expect a balanced training dataset to get the desired result. With this motivation, we have come up with the following methodology to complement the abundance of publicly available claim samples with non-claim samples.

We have created a large scale dataset of non-claim sentences and made it publicly available at the following Github repository [1]. Although such a large dataset is not needed for the transfer learning based models, we aim to assist future researchers in training simpler models which would expect a balanced ratio of classes.

These non-claim sentences have been obtained from *Wikipedia*. Wikipedia's citation policy states that;

> "Wikipedia's verifiability policy requires inline citations for any material challenged or likely to be challenged, and for all quotations, anywhere in article space."

The definition of a claim also happens to be;

> "A statement about the world that can be verified".

Since *Wikipedia* expects citations for anything verifiable, it inturn requisites that claims in their articles be cited. *Wikipedia* also expects quotations to be cited which may or may not be claims. Thus sentences without citations would be non-claims and sentences with citations can be both. By filtering out citations from Wikipedia articles we would be left with only non-claims. Since any user can edit a *Wikipedia* page, and it can happen that beginners are not entirely aware of its policy and ignore these rules. We only work with pages having pending changes protection, extended confirmed protection, semi-protection and full protection. Only verified users can edit these pages with these protection levels.

We created a web crawler to access Wikipedia's page contents and filter out the sentences containing citations leaving us with only non-claim sentences. This crawler is programmed using Python and uses libraries such as Spacy, Regex and Wikipedia's API features. For our final balanced dataset, we use the samples returned by the above-mentioned crawler as non-claims and take data samples of class claim from *FEVER* and Wang (2017). We summarize some of the statistics of our final combined dataset in Table 2. Moreover, some samples from the combined dataset are presented in Table 3.

4.2 Baseline model

We trained a two-layered Long short-term memory *(LSTM)* network (Hochreiter and Schmidhuber, 1997) as a baseline model. Its input words are converted into word embeddings using google's pre-trained *word2vec* model (Mikolov et al., 2013), which creates word embeddings of dimension 300. The network's configurations consist of a hidden layer size of 300, a learning rate of 0.001, Adam Optimizer (Kingma and Ba, 2015), negative log-likelihood as loss function and a mini-batch size of 64 for training. We train for the dataset mentioned in section 4.1, in a five-fold cross-validation set manner, on a Tesla GPU based computing cluster.

[1] https://github.com/ashish6630/Claim_extraction.git

Sentence	Claim
In Georgia, women earn 82 cents for every dollar earned by men.	Yes
Bermuda Triangle is in the western part of the Himalayas.	Yes
In Azerbaijan 53% of the population, according to polls, state that religion has little to no importance in their lives.	Yes
Tupac Shakur is my favourite Rapper.	No
Some praised Rogan for hosting a pragmatic discussion while others seemed rather stunned by Sanders decision to appear on the show at all.	No

Table 3: Some samples from the proposed dataset

The results are summarized in Table 4.

Further experiments were tried with a varying range of dropout and learning rates, but there was no increase in the scores. Increasing the size of the training data is the only option to improve model accuracy. The transfer learning based model is described in the next section.

4.3 Transfer learning models

We perform transfer learning by fine-tuning a pre-trained *BERT* base and *DistilBert* base model. These models have the advantage of having been pre-trained on a large corpus for the unsupervised task of language modelling. They also include multiple self-attention heads which encode how a word in a sentence relates to the other words, and this information can prove useful during the final classification.

Although *BERT* does not use character level embedding like *ELMO*, it is still able to avoid out of vocabulary errors by breaking words into subwords wherever possible. Unlike the LSTM, where the next time step of the computation requires information from the previous time step, making it challenging to parallelize, *BERT* being a transformer-based model processes the entire sentence at once. Training time is now significantly faster. The sentence embedding generated at the end is used for further steps, and the rest of the output is discarded. This embedding is passed through a linear transformation layer to map the embedding of size 768 to the class size of 2, i.e. claim and non-claim. These models were fine-tuned on our combined dataset shown in Table 3 for three epochs with a learning rate of 2e-5 with *Adam* optimizer on a Tesla GPU cluster. Table 4 contains the results after training for three epochs with 2000 samples (1000 claims and 1000 non-claims) and testing with the remaining samples from the dataset mentioned in section 4.1.

The ratio of the distribution of claims can vary depending on the scenario, e.g. A presidential debate transcript will mostly consist of claims. In contrast, some scenarios might only have a lesser percentage of claims. Taking that into account, we experimented with varying ratios of claims and non-claims, and the results are shown in Table 4. The results of the transfer learning model are robust regardless of the ratio of claims in the dataset. As highlighted in the table, both *BERT* and *DistilBert* obtain promising results with only a fraction of the dataset.

5 Conclusion and Future Work

In this work, we have presented our publicly available dataset and quantified the performance of *BERT* and *DistilBert* in detecting claims in general. These are one of the most advanced transfer learning methods available and can provide highly accurate results with fewer data. The transfer learning based pre-training of these models helped it to achieve high evaluation scores despite having been trained with a fraction of the available dataset.

Until now, Fake news detection has been thought of as a two-step process consisting of detecting claims and verifying them. The first part can be further subdivided into detecting claims and sorting them according to the check worthiness of a claim. Other research domains such as argument mining would benefit from this since they would want to sort claims according to argumentative claims rather than check worthiness. Researchers can thus decide themselves the type of claim to filter out. We will address the problem of check worthiness in our future work.

Model	Data distribution		Accuracy	Precision	Recall	F1
	% of claims	% of non-claims				
LSTM(Baseline)	50%	50%	70.32%	74.37%	77.82%	76.06%
BERT	10%	90%	95.32%	95.12%	96.24%	**95.68**%
BERT	25%	75%	96.13%	96.43%	96.89%	**96.66**%
BERT	50%	50%	98.42%	97.38%	98.61%	**97.99**%
BERT	75%	25%	97.06%	97.10%	97.79%	**97.44**%
BERT	90%	10%	95.54%	95.21%	95.88%	**95.54**%
DistilBert	10%	90%	95.36%	94.95%	95.39%	95.17%
DistilBert	25%	75%	95.85%	95.72%	95.91%	95.81%
DistilBert	50%	50%	97.78%	96.61%	98.37%	97.66%
DistilBert	75%	25%	96.12%	96.24%	96.31%	96.27%
DistilBert	90%	10%	94.47%	94.59%	94.63%	94.61%

Table 4: Final Results on the proposed dataset

References

Ben Adler and Giacomo Boscaini-Gilroy. 2019. Real-time claim detection from news articles and retrieval of semantically-similar factchecks. In *Proceedings of the Third International Workshop on Recent Trends in News Information Retrieval, co-located with 42nd International ACM Conference on Research and Development in Information Retrieval (SIGIR 2019), Paris, France, July 25, 2019*, volume 2411 of *CEUR Workshop Proceedings*, pages 36–41. CEUR-WS.org.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255. IEEE Computer Society.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Naeemul Hassan, Fatma Arslan, Chengkai Li, and Mark Tremayne. 2017. Toward automated fact-checking: Detecting check-worthy factual claims by claimbuster. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pages 1803–1812. ACM.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 328–339. Association for Computational Linguistics.

Israa Jaradat, Pepa Gencheva, Alberto Barrón-Cedeño, Lluís Màrquez, and Preslav Nakov. 2018. ClaimRank: Detecting check-worthy claims in Arabic and English. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 26–30, New Orleans, Louisiana. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Lev Konstantinovskiy, Oliver Price, Mevan Babakar, and Arkaitz Zubiaga. 2018. Towards automated factchecking: Developing an annotation schema and benchmark for consistent automated claim detection. *CoRR*, abs/1809.08193.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing LSTM language models. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Victor SANH, Lysandre DEBUT, Julien CHAUMOND, Thomas WOLF, and Hugging Face. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 809–819. Association for Computational Linguistics.

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the Workshop on Language Technologies and Computational Social Science@ACL 2014, Baltimore, MD, USA, June 26, 2014*, pages 18–22. Association for Computational Linguistics.

William Yang Wang. 2017. "liar, liar pants on fire": A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*, pages 422–426. Association for Computational Linguistics.

Assamese Word Sense Disambiguation using Genetic Algorithm

Arjun Gogoi
Dibrugarh University
Dibrugarh-786004, India
agogoi440@gmail.com

Nomi Baruah
Dibrugarh University
Dibrugarh-786004, India
baruahnomi@gmail.com

Shikhar Kr. Sarma
Gauhati University
Guwahati-781014,India
sks001@gmail.com

Abstract

Word sense disambiguation (WSD) is a problem to determine a word according to the context in which it occurs. There are plenty amount of works done in WSD for some languages such as English, but research work on Assamese WSD remains limited. It is a more exigent task because Assamese has an intrinsic complexity in its writing structure and ambiguity, such as syntactic, semantic, and anaphoric ambiguity levels. A novel unsupervised genetic word sense disambiguation algorithm is proposed in this paper. The algorithm first uses WordNet to extract all possible senses for a given ambiguous word, then a genetic algorithm is used taking Wu-Palmer's similarity measure as the fitness function and calculating the similarity measure for all extracted senses. The winner sense which will have the highest score declared as the winner sense.

1 Introduction

The specific task that resolves the lexical ambiguity is commonly referred to as Word Sense Disambiguation(WSD). It is a process to assign a meaning to a word based on the context in which it occurs. WSD is quite important for resolving some problems in such NLP applications as text clustering and classification, word similarity computation, and so on.

Suppose there is a word জিকা/Jika. The word জিকা/Jika is an ambiguous word which has multiple senses.
Sense 1: জিকা/Jika= To Win
Sense 2: লতাজতীয় গছত লগা তৰকাৰী হিচাপে খোৱা এবিধ দীঘল আৰু পাতল ফল /Lota jatio gosot loga torkari hisape khuwa abhid dighol aru patol fol = a kind of Vegetable
A. মায়ে আজি জিকাৰ তৰকাৰী ৰান্ধিছে/Maa'e aaji jikar torkari randhise/Mother is cooking jika curry today.
→ This জিকা/Jika gives a sense as a kind of vegetable
B.ৰাহুলে বাজী জিকিল/Rahul e baji jikil/Rahul won the toss
→ This জিকা/Jika gives a sense as to win.

Various WSD Methods uses external knowledge source, for example, Word Net/ Thesaurus as a basement to disambiguate the ambiguous words. There are also some hybrid methods, such as an Iterative Approach to WSD, Sense Learner which first used textual definitions of senses from a machine-readable dictionary and then used a corpus to identify and calculate the relatedness between two senses. Another kind of approach to disambiguate words using machine learning-based algorithms (Tatar, 2005) rather than taking it directly from an explicit knowledge source such as:

i. Supervised WSD: These types of approaches use machine learning techniques and sense annotated corpora to train the data.
ii. Unsupervised WSD: These types of methods are based on unlabeled corpora, and make groupings of similar words based on their similarity.

In this paper, we propose to do word sense disambiguation using genetic algorithms which is a novel unsupervised based algorithm. The basic idea is to maximize the overall similarity of disambiguated senses. We propose a novel similarity measure which combines domain information with the Wu-Palmer similarity measure (Zhang et al., 2008) to

Proceedings of the 17th International Conference on Natural Language Processing, pages 303–307
Patna, India, December 18 - 21, 2020.

calculate the similarity between senses in Word-net.

The structure of this paper is as follows: Related works of the proposed approach, Methodology of the proposed approach, Data set and Experiments of the proposed approach, Some close observations, Conclusion of the proposed approach.

2 Related works

As no work on the unsupervised approach is done to date in the Assamese language, we had considered work in Indian languages done to date as Assamese is one of the Indian languages. Most of the Indian language structure is the subject-object-verb(sov). The proposed works in the unsupervised approach are as follows:

A work on Malayalam language WSD have proposed by (Sankar et al .,2016) . In this paper, an Unsupervised Approach has been proposed and the proposed system makes use of a corpus which is taken from various Malayalam web documents. Based on the similarity between the given input and the sense clusters, the most similar sense is selected as the winner sense. The proposed system gives an accuracy of 72 percent.

A work on Gujarati language WSD have proposed by (Vaishnav and Sajja,2019) . In this paper, a Genetic algorithm has been proposed which uses a Knowledge-based approach where Indo-Aryan WordNet for Gujarati is used as a lexical database for WSD .

A work on Hindi language WSD have proposed by (Tayal et al., 2015). In this paper, an Unsupervised Approach has been proposed and applied the Fuzzy C-Means Clustering algorithm to form clusters. They test the approach on the corpus created using Hindi news articles and Wikipedia and then compare the approach with other methods available in all the previous works done for the Hindi Language. The training data used consists of 3753 words in total and found an accuracy of approximately79.16 percentage .

A work on Hindi language WSD have proposed by (Sangwan and Singh, 2013). In this paper, a Genetic algorithm has been proposed and this is the very first use of a Genetic Algorithm for the Hindi language WSD. Here, Hindi Wordnet has been used as a lexical database for WSD. To date, they have found some results which are under process.

A work on Gujarati language WSD have proposed by (Mamulkar and Nandanwar, 2020) have proposed "Word Sense Disambiguation for the Marathi language". In this paper, they are proposing the Genetic Algorithm Approach for the disambiguation of ambiguous words. In their survey, it is found that a genetic algorithm gives better results than other approaches

A work on Tamil language WSD have proposed by (Priya et al .,2018). In this paper, the Hierarchical Fuzzy Clustering Algorithm is applied along with the WordNet dictionary. To identify the disambiguated words, sense identification is performed for the adjectives, and comparison is performed. On comparing with previous methods for WSD in the Tamil language, their work gives a better result with a percentage of 91.2 percentage .

A work on Bengali language WSD have proposed by (Pal and Saha, 2019). In this paper, Word Sense Disambiguation has been done using the unsupervised methodology. In this work, clustering has been implemented using the weka tool and the test sentences are extracted from the Bengali text corpus developed in the TDIL (Technology Development for Indian Language) project of the Govt. of India.

A work on Assamese language WSD have proposed by (Sarmah and Sarma,2016) using a supervised Machine Learning approach "Decision Tree-based". In this approach, a few polysemous words with different real occurrences in Assamese text with manual sense annotation were collected as the training and test dataset. They have been able to get an average F-measure of .611 when 10-fold cross-validation evaluation when performed on 10 Assamese ambiguous words.

A work on Assamese language WSD have proposed by (Borah et al., 2014). This is the first work to the best of our knowledge on developing an automatic WSD system for the Assamese language using Naive Bayes approach. They have conducted the experiment

using 25 highly polysemic words using the articles collected from the Assamese textbook of class X.

A work on Assamese language WSD have proposed by (Kalita and Barman, 2020) in which the Walker algorithm, a thesaurus based algorithm, is applied that makes use of the category of each word as defined in the thesaurus. But for a language like Assamese, WSD is far from being a topical approach. Moreover, due to the non-existing thesaurus with a tagged category particularly for the Assamese language.

3 Methodology

The initial idea of Genetic Word Sense Disambiguation(GWSD) comes from disambiguating a set of domain words extracted from a domain corpus. Because the terms can belong to the same domain and we can assume that they tend to be similar in semantics with each other. So a good disambiguation solution should be a solution that assigns one sense to each term and increases the overall similarity on the set of selected senses. Each sense of a word corresponds to a synset present in the WordNet and hence we use genetic algorithms to find an optimal set of senses, which increases the overall similarity as it is a global search heuristic. For the WSD task, in a given population each chromosome which is equal to the all possible sense for a given term in the WordNet will be tested using a fitness function, and the most fitted chromosome or sense will be declared as the correct sense.

We give a simple example of initialization. If we want to disambiguate a word, the population size is equaled to the collection of all the possible senses from the wordnet, and for each sense, we calculate the fitness function, and the fittest sense is declared as the winner sense i.e which sense scores the maximum similarity using the fitness function. A fitness function is an objective function that is used to summarize, as a single figure of merit, how close a given design solution is to achieving the set aims. Fitness functions are used in genetic programming and genetic algorithms to guide simulations towards optimal design solutions. In our proposed approach, we are using conceptual similarity between two senses using Wu-Palmer's similarity measure as the fitness function as described in section 3.1.

For example: ভেল/Bhul/Vegetable.

There a total of 2 sense for ভেল/Bhul/Vegetable so 2 will be the population size and then we calculate the fitness function as mentioned above.

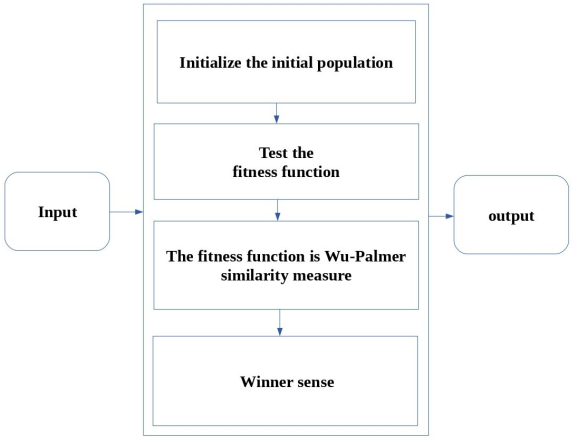


Figure 1: Diagram of the proposed approach

In Figure 1, after we take the input at first initialize the initial population i.e collect all the possible senses from the wordnet. For each chromosome/Sense in the population, test the fitness function(Wu-Palmer's similarity measure). The sense/ chromosome which gets the maximum similarity to assign declared as the winner sense i.e required output.

3.1 Conceptual similarity

In our approach, the conceptual similarity between two senses is measured using Wu-Palmer's similarity measure (Zhang et al. , 2008). In Wu-Palmer's similarity measure, Conceptual similarity between a pair of terms C1 and C2 in a hierarchy is defined by (Slimani, 2013) as:

$$SIM(C1, C2) = (2 \times N/(N1 + N2 + 2 * N)) \quad (1)$$

Where N1 and N2 are the distance that separates, respectively, the concept C1 and C2 from the specific common concept and

N is the distance which separates the closest common ancestor of C1 and C2 from the root node.

In our proposed approach, at first to disambiguate a word from an input sentence we draw a tree based on their similarity measure considering the Syn-sets(meaning) from the WordNet and calculate its Wu-Palmer's similarity measure.

For example. Wu-Palmer's similarity measure for the two senses জ্বৰ/Jor/Fever and বেমাৰ/
Bemar/Illness) is given by
SIM(জ্বৰ/Jor/Fever,বেমাৰ/Bemar/Illness)=1.2

4 Data set and Experiment

We evaluate the genetic algorithm developed and applied a sense annotated Assamese corpus as shown in the table below. The sense inventory used in this research has been derived from the Assamese WordNet. There are a total of 50000 words and among them, 15606 words are ambiguous (Baruah et al., 2020).

Table 1: Details of the corpus

Details	Values
Total Number of Words	50001
Ambiguous Words(Nouns)	15606
Total Number of Unique Words	12282
Total number of Instances	12282
Total number of word senses	50001
Total number of instances per word	4.07
Total number of senses per word	1.0

Using precision, recall metrics we have done the experiment for genetic algorithm as follows:

$$Precision = correct\ senses/Sentences\ taken \quad (2)$$

$$Recall = correct\ senses/Total\ Sentences\ taken \quad (3)$$

Table 2: Results

Precision	81.25
Recall	74.28

As no unsupervised works are done earlier, our work is compared with the available works related to supervised approaches in the Assamese language.Though we have been able to get a few papers only which have calculated their results in precision and recall metrics, we have been able to get quite better results while comparing.

5 Some close observations

a.Very short sentence: Having sentences too short in length, the proposed system could not retrieve sufficient data and it creates difficulty in the case of the similarity measure.
b.Spelling error: It is a very important factor as spelling errors in words can decrease the performance of the system.
c.Scarcity of information in Assamese WordNet: In this dictionary, synonymous sense definitions of the common Assamese ambiguous words are absent and it is a great difficulty in the proposed approach.
d.Same contextual words with different senses: various sentences that are not similar through their similar contextual words. For example:

I. মায়ে আজি ভোলৰ চব্জি ৰান্ধিছে/Maa'e aaji bhulor sobji randhise/Mother is cooking bhul curry today.
This ভোল/Bhul/Vegetable gives a sense as a kind of Vegetable
II.এইটো মুৰ ভুল/Aitu mur bhul/This is my fault.
This ভোল gives a sense as To do something wrong.

6 Conclusion

In this paper, the WSD system for the Assamese language using a genetic algorithm has been proposed and analyzed. Works on Assamese language using genetic algorithm

are almost none. Therefore we have an attempt to disambiguate the ambiguous words using a genetic algorithm. This proposed system has been tested on a manually sense-tagged corpus of 30 most ambiguous words. Wu-Palmer similarity measure method has also been applied as a fitness function to the algorithm and found that Precision is 81.25 percentage and Recall is 74.28 percentage.

In the future, the scalability of the proposed system can be improved by adding more ambiguous words to the Assamese language. This proposed system is one target word WSD and can be extended to an all-word WSD system and will show more good results once the Assamese will fully complete.

7 Refrences

Deborah Gail Tatar. 2005. Word Sense Disambiguation by Machine Learning Approach. *Fundamental Informaticae*, pages 433-442.

ChunHui Zhang, Yiming Zhou, and Trevor Martin. 2008. Genetic Word Sense Disambiguation Algorithm. *Second International Symposium on Intelligent Information Technology Application*, pages 123-127.

Sruthi Sankar, Reghu Raj, Jayan. 2016. Unsupervised Approach to Word Sense Disambiguation in Malayalam. In textitproceedings of the International Conference on Emerging Trends in Engineering, Science and Technology, pages 1507 – 1513.

Zankhana Vaishnav and Priti Sajja. 2019. KnowledgeBased Approach for Word Sense Disambiguation Using Genetic Algorithm for Gujarati. *Information and Communication Technology for Intelligent Systems*, pages 485-45.

Devendra Tayal, Leena Ahuja, Shreya Chhabra. 2015. Word Sense Disambiguation in Hindi Language Using Hyperspace Analogue to Language and Fuzzy C-Means Clustering. In *Proceedings of the 12th International Conference on Natural Language Processing*, pages 49–58.

Shabnam Sangwan and Paramjit Singh. 2013. Genetic Algorithm Based Hindi word sense disambiguation. *International Journal of Computer Science and Mobile Computing* , 2(5):139-144.

Kalyani Mamulkar and Lokesh Nandanwar. 2020. Marathi Word Sense Disambiguation Using Genetic Algorithm. In *IJACEN* , pages 16-18.

Mohana Priya, Krishna Priya, Geetha Harini, Ragavi. 2018. Handling WSD using Fuzzy Hierarchical Clustering of Sentence Level Text. *International Journal of Research in Advent Technology*, 6(11):3209 -3214.

Alok Ranjan Pal and Diganta Saha. 2019. Word Sense Disambiguation in Bengali language using unsupervised methodology with modifications. *Sådhanå*, pages 44:168.

Jumi Sarmah and Shikhar Kumar Sarma. 2016. Decision Tree based Supervised Word Sense Disambiguation for Assamese. *International Journal of Computer Applications*, 141(1):42-48.

Pranjal Protim Borah, Gitimoni Talukdar, Arup Baruah. 2014. Assamese Word Sense Disambiguation using Supervised Learning. *International Conference on Contemporary Computing and Informatics*, pages 946-950.

Purabi Kalita and Anup Kumar Barman. 2015. Implementation of Walker algorithm in Word Sense Disambiguation for Assamese language. In *ISACC*, pages 136-140.

Thabet Slimani. 2013. Description and Evaluation of Semantic Similarity Measures Approaches. *International Journal of Computer Applications*, 80(10):25-33.

Nomi Baruah, Arjun Gogoi, Shikhar kumar Sarma, Randeep Borah. 2020. Utilizing corpus statistics for Assamese word sense disambiguation. *CoCoNET'2020.*

Free Word Order in Sanskrit and Well-nestedness

Sanal Vikram and **Amba Kulkarni**
sanal.vikram@gmail.com, ambakulkarni@uohyd.ac.in
Department of Sanskrit Studies, University of Hyderabad

Abstract

The common wisdom about Sanskrit is that it is free word order language. This word order poses challenges such as handling non-projectivity in parsing. The earlier works on the word order of Sanskrit have shown that there are syntactic structures in Sanskrit which cannot be covered under even the non-planarity. In this paper, we study these structures further to investigate if they can fall under well-nestedness or not. A small manually tagged corpus of the verses of Śrīmad-Bhagavad-Gītā (BhG) was considered for this study. It was noticed that there are as many well-nested trees as there are ill-nested ones. From the linguistic point of view, we could get a list of relations that are involved in the planarity violations. All these relations had one thing in common—that they have unilateral expectancy. It was this loose binding, as against the mutual expectancy with certain other relations, that allowed them to cross the phrasal boundaries.

1 Introduction

Sanskrit is inflectionally rich and it rarely uses position to encode any syntactic or semantic relation between words. This enables Sanskrit to move the words freely in a sentence. Within Indian tradition, the word order was regarded as free, provided the proximity (*sannidhi*) is not violated. The Indian theorists found the sentences with different arrangements of words to be equivalent in meaning, with an exception of subject-predicate (*uddeśya-vidheya*). The difficulty in understanding the verses due to deviation from the 'default' word order, however, had been realised. The commentators commenting on the verses have followed this 'default' word order known as the *anvaya* of a verse (*śloka*)[1]. But this default word order is not followed strictly either by the commentators or by the authors while composing prose. When the Sanskrit texts started being translated into fixed word order European languages, the free word order of Sanskrit had been noted. The westerners tried to propose a framework for the Sanskrit word order. However, these studies also lead to only a preferential, or frequent arrangement, and not 'the' arrangement. The deviations were considered to be the exceptions. Even while discussing the syntax, much emphasis had been laid on the concord and government rather than order.

Though Sanskrit allows free movement of words, and there are preferential word orders, certainly not all permutations of the words are allowed. So the attention of the researchers then was drawn to the restrictions on word order rather than possible word orders. The first systematic account of the word order in Sanskrit was by Staal (1967). He introduced a concept of 'wild tree' which allows the movement of the items within a phrase freely. In this model, any two items from different phrases cannot interleave or the words in one phrase cannot leave their place and move to the domain of other phrases.

This model was formalised and empirically tested by Gillon on a corpus of approximately thousand Classical Sanskrit prose sentences. Half of the sentences of this corpus was from a single text,[2] and the other half was selected at random from the sentences found in Apte (1925). His empirical observations confirm the model suggested by Staal

[1] *ādau kartṛpadam vācyam dvidīyādipadam tataḥ | ktvātumunlyap ca madhye tu kuryād ante kriyāpadam || viśeṣaṇam puraskṛtya viśeṣyam tadanantaram | kartṛkarmakriyāyuktam etad anvayalakṣaṇam ||*
(Samāsacakra karikās 4, 10)
(English: Starting with *kartṛ*, followed by other words, placing the non-finite verb forms such as *ktvā*, *tumun*, *lyap* in between, place the main verb at the end. Starting with adjectives, targeting the headword, in the order of kartṛ-karma-kriya gives an *anvaya* (natural order of words).)

[2] Prose commentary by Dharmakīrti on the *Pramāṇavārttika*.

Proceedings of the 17th International Conference on Natural Language Processing, pages 308–316
Patna, India, December 18 - 21, 2020.

with an exception of movement of genitives and adjectives across the clause boundaries.

Aralikatti (1991) studies the modern Sanskrit texts and conversations from the point of view of the flexibility in word order. He found that the modern writings and the conversations follow the default word order. Scharf et al. (2015) presents the preliminary observations with regards to the comparison of the word order in prose and verse which confirm more flexibility in verses than in prose.

In order to develop a computational parser for Sanskrit, these theoretical insights are very much useful. Kulkarni et al. (2015) studied and carried out an empirical study of the verses in Srimad-Bhagavad-Gītā(BhG). The aim of this study was to gain insights regarding the flexibility in the word order to build a computational model of grammatical sentences in Sanskrit. They could fit a weakly non-projective (or planar) model for the Sanskrit sentences, barring a few cases. One important observation was that the number of cases of violation of planarity condition in verse was higher than the number of exceptions studied by Gillon in prose. Another observation was with respect to the relations involved in the planarity violation. It was observed that in addition to the two relations viz. the adjective (*viśeṣaṇa*) and the genitive, pointed out by Gillon, a few other relations such as vocative, negation etc. also violated the planarity of the graph. But these relations were not as frequent as the genitive and adjective. At the same time, the behaviour of these relations was the same irrespective of whether the text is a prose or a verse.

These relations which do not conform to the planar graphs had a special status. A peep into the Indian theories of verbal cognition revealed that these exceptions correspond to the cases where the words have unilateral expectancy (*utthāpya ākāṅkṣā*). Such grammatically accepted sentences were studied further in order to build a proper computational model for parsing them. In this paper, we test whether the exceptions to the planar graphs fall under the category of well-nested graphs or not.

The organisation of the rest of the paper is as follows. In the next section, we describe various parameters to classify the syntactic structures mathematically. In the third section we discuss two major concepts—the concept of expectancy (*ākāṅkṣā*) and the concept of proximity (*sannidhi*)—from Indian theories of verbal cognition (*śābdabodha*) that are useful from the point of view of dependency. In the fourth section, we describe the empirical experiments we carried out to classify the cases of proximity violation. We show that the violations do occur both in the well-nested as well as ill-nested graphs and that the non-planarity is mainly due to the adjectival and genitive relations with a few cases of other non-*kāraka* relations such as negation, vocative, conjunction, etc. Finally we conclude that the *utthita* and *utthāpya ākāṅkṣā* provide a better classification for the non-planar graphs rather than well-nested and ill-nested classification.

2 Dependency Structures

In this section we present the formal definition of various mathematical structures associated with the dependency. The dependency parse of a sentence is expressed in the form of a tree structure. This tree is a Directed Acyclic Graph with one root node, and all other nodes connected to at least one other node in the tree by a direct edge.

2.1 Projectivity Principle

The principle of projectivity imposes certain constraints on the dependency tree which bans certain dependency structures.

> A sentence is projective if and only if we can draw a dependency tree whose every node can be projected by a vertical line onto its word form in the surface string without crossing another projection or a dependency edge.

Thus, if A depends directly on B and some other element C intervenes between them (in linear order of string), then C depends directly on A or on B or on some other intervening element. Thus the projectivity requires the node to dominate a continuous substring of the sentence and bans on discontinuous constituents. The intuitive 'wild tree' notion of Staal comes very close to this projectivity principle. Dependency grammars that allow only projective structures are closely related to the context-free grammars (Gaifman, 1965), and hence can be parsed in cubic time (Eisner, 1996).

It was noticed that there are certain constructions in natural languages that do not fit in with the dependency tree satisfying the projectivity principle. Hence the constraint was further relaxed, so as to allow some non-projectivity. Figure 1 shows one dependency graph exhibiting non-projectivity.

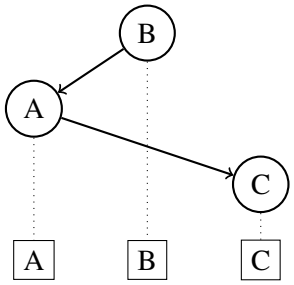


Figure 1: Non Projective Graph

2.2 Weak Non-projectivity (Planarity)

A dependency tree is weakly non-projective if there is no crossing among the relation edges when plotted on one side of the sentence (either above or below). This is also a planar graph. The planar graph in computational linguistics is the same as the one from the computer science with one modification that all the edges are drawn on the same side (either above or below) of a sentence. Thus a dependency graph is planar, if it does not contain nodes i, j, k, l such that $i < j < k < l$ and $edge(i, k)$ and $edge(j, l)$.

For example, Figure 1 shows the crossing of a dependency relation with the projection. But the same dependency relations when drawn with words arranged in a linear order and the edges drawn above the sentence, the crossing disappears (See Figure 2).

Figure 2: Planar dependency graph for (1)

2.3 Well-nestedness

The well-nested constraint imposes restrictions on the positioning of the disjoint sub-trees. Two trees are called disjoint if neither of their roots dominates the other. Two subtrees T_1 and T_2 interleave, iff there are nodes l_1, r_1 of T_1 and l_2, r_2 of T_2, such that $l_1 < l_2 < r_1 < r_2$. A dependency graph is well-nested, iff no two of its disjoint subtrees interleave. If the two trees that interleave are not disjoint, that is if the root of one tree governs the root of the other tree, then the interleaving of edges in such trees is allowed. Dependency trees which have overlapping edges across disjoint subtrees are considered as ill-nested.

Figure 3 is well-nested, because the edges $1 \rightarrow 3$ and $2 \rightarrow 5$, of the two disjoint trees $1 \rightarrow 3 \rightarrow 4$ and $2 \rightarrow 5$, interleave, but the node 1 of the first tree governs the node 2 of the second tree.

In Figure 4, the edges $3 \rightarrow 5$ and $2 \rightarrow 4$ interleave, and neither 2 nor 3 is governed by the other. Hence this is an ill-nested tree.

In Figure 5, the edges $1 \rightarrow 3$ and $2 \rightarrow 5$ interleave, and again neither 1 nor 2 govern the other. Hence this is also an ill-nested graph.

All projective trees are weakly non-projectives and all weakly non-projective trees are well-nested by definition.

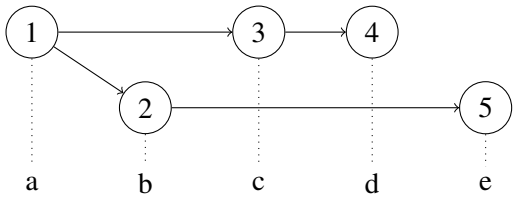


Figure 3: Well-nested graph

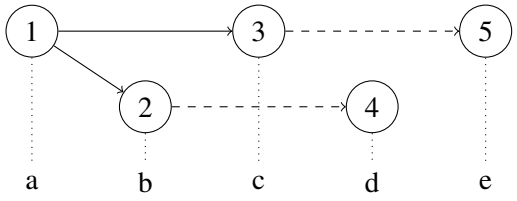


Figure 4: Ill-nested graph

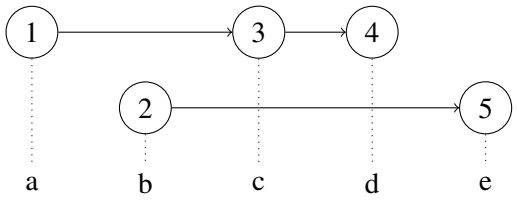


Figure 5: Ill-nested graph

In the next section, we look at some linguistic concepts that are useful for understanding the potency of various dependency relations in the interleaving.

3 Indian Theories

According to the exegesists (*mīmāṃsakas*) a sentence is defined as

> *arthaikatvāt ekam vākyam*
> *sākāṁkṣam cet vibhāge syāt | MS 2.1.46*

(A sentence is an integral unit conveying a single purpose, and when it is split into two parts, some words in one part would have an expectancy for

some other words in the other part.)
This implies that each word in a sentence either satisfies an expectancy of or has an expectancy for some other word in a sentence. That is every word in a sentence should be connected with at least one other word in it. Let us represent the words in a sentence by the nodes, and the expectancies between words by edges joining the nodes. Two nodes connected by an edge do not have the same status. One of them has an expectancy and the other one satisfies the expectancy. Hence we use directed edges to mark this asymmetry.

3.1 Expectancy and Proximity

Indian grammatical texts discuss two kinds of expectancies—*utthita* and *utthāpya*. The expectancies which are mutual, direct and natural are termed *niyata/utthita ākāṅkṣā* (restricted or risen expectancy) (Kunjunni Raja, 1963). The expectancy between a verb and the words denoting kārakas or between relational words falls under this category. In a Sanskrit sentence '*dvāram pidhehi*' (close the door), the verb 'close' has an expanctancy of a *karma* (object) which is fulfilled by the word '*dvāram*' (the door). Inversely, a verb is expected with the word 'door' mentioning what to do with the door. Expectancies of *kāraka* relations are mutual. In contrast to mutual expectancy, the expectancy that is unilateral is called *aniyata* or *utthāpya ākāṅkṣā* (unrestricted or to be raised). This is aroused only if necessary. So it is potential. For example, in a phrase such as 'white cow', the expectancy of 'white' for a noun is natural, but the expectancy of 'cow' to have an adjective is potential. It gets aroused only in the presence of an adjective such as 'white'. Even a noun in apposition may arouse an expectancy. Similarly, a noun in genitive arouses an expectancy of another noun. And this expectancy is uni-directional and not mutual.

Another concept from the Indian grammar viz. *sannidhi* (proximity) puts certain constraints on the word order. It states that the words that are related to each other should not be intervened by other words.

The proximity, along with the expectancy was further studied by Kulkarni et al. (2015). They carried out an empirical evaluation of a manually annotated corpus to understand the nature of this ban on crossing of dependency relations. They found that one of the relations involved in the crossing of edges was corresponding to the unilateral expectancy. A few cases were also found where both the relations involved had mutual expectancies.

In this paper, we study these cases where the planarity constraint is violated and investigate if these cases of violations are well-nested or not.

4 Dependency Graphs and Planarity

Sanskrit is inflectionally rich. So the common wisdom is that we can move around the words in any order. For example, the following sentence with three words,

(1) *śvetaḥ aśvaḥ dhāvati*
White horse runs

can have 3! (=6) permutations. But among these the following permutation, for example,

(2) *aśvaḥ dhāvati śvetaḥ.*
A_horse runs white.

is non-projective (See Figure 6).

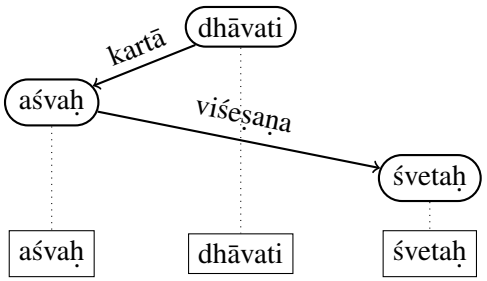


Figure 6: Non Planar Graph

However if the relation edges are plotted above the sentence, we notice that it produces a planar graph (See Figure 7).

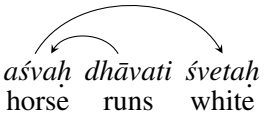


Figure 7: Planar dependency graph for (2)

But every non-projective graph may not produce a planar graph. For example, consider sentence (3).

(3) *Rāmaḥ dugdham pītvā*
Rama{nom.} milk{acc.} drink{abs.}
śālām gacchati.
school{acc.} go{3p.sg.}.

Rama goes to school after drinking milk.

This sentence has 5 words. But not all the 5! (=120) combinations are meaningful. The following sentence obtained by permuting the words in the above sentence is not meaningful.

(4) **Rāmaḥ* *śālām* *dugdham*
Rama{nom.} school{acc.} milk{acc.}
gacchati *pītvā.*
go{3p.sg.} drink{abs.}.

*Rama to school milk goes drinking.

It not only violates the projectivity principle, but even the graph is non-planar as there are crossings (See Figure 8). And this sentence is grammatically ill-formed.

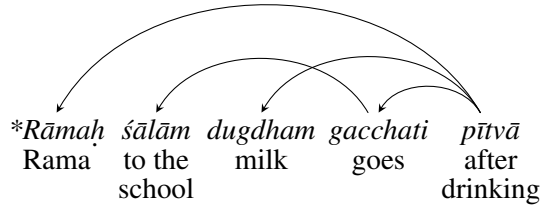


Figure 8: Planar dependency graph for (4)

On the other hand, the sentence (5) below has a non-planar graph and the sentence is grammatically well-formed.

(5) *eṣaḥ vāk-viṣaya-bhūtaḥ saḥ te vīraḥ.*
This speech-topic-become he your hero.

This is the hero who has become the topic of your speech.

In this sentence, the demonstrative adjective '*saḥ*' modifying a predicate noun '*vīraḥ*', intervenes between its predicate '*bhūtaḥ*' and the agent (*kartā*) of the 'speech' (*vāk*) viz. '*te*', as shown in Figure 9.

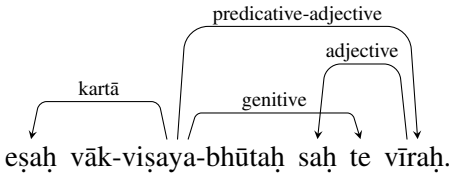


Figure 9: Planar dependency graph for (5)

Below is a part of a verse from Śrīmad-Bhagavad-Gītā (BhG) that exhibites similar phenomenon.

cañcalam hi manaḥ kṛṣṇa
pramāthi balavat dṛḍham |
tasya aham nigraham manye
vāyoḥ iva suduṣkaram || BhG 6.34

(English: For, O Krishna, the mind is unsteady, turbulent, strong and obstinate, I consider its control to be as difficult as of the wind.)

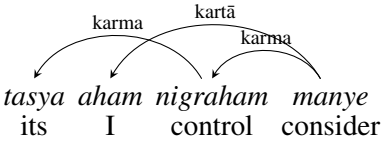


Figure 10: Analysis of BhG 6.34

In the second line of this verse the main verb is '*manye*' (consider) whose *kartā* is '*aham*' (I). The *karma* of the verbal noun '*nigraham*' (control) is the pronominal '*tasya*' (its), which refers to '*manaḥ*' (mind) in the first part of the verse. Thus the word sequence '*tasya aham nigraham manye*' produces two crossing edges involving the relations of *kartā* and *karma*.

Let us see one more example. This is 18th śloka of 10th chapter.

vistareṇa ātmano yogaṁ
vibhūtiṁ ca janārdana |
bhūyaḥ kathaya tṛptiḥ hi
śṛṇvato nāsti me'mṛtam || BhG 10.18

(English: O Janardan, tell me again elaborately your own yoga and manifestations. For, I'm not satisfied when I listen to your immortal words.)

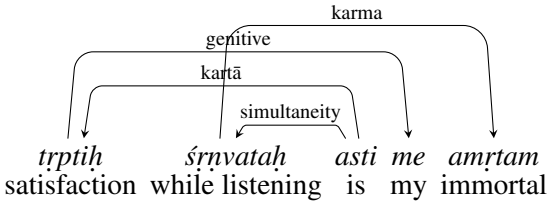


Figure 11: Analysis of BhG 10.18

In this verse, again we look at the second part of the second line in the verse. The *kartā* of the main verb '*asti*' (is) is '*tṛptiḥ*' (satisfaction), the *karma* of '*śṛṇvataḥ*' (while listening) is '*amṛtam*' (immortal), and there is a genitive relation between '*me*' (my) and '*tṛptiḥ*' (satisfaction). We see two crossings, one between the *kartā* and *karma*, and the other between genitive and *karma*.

There is an important difference between the crossing in Figures 9, 10 and 11 though all of them are grammatically sound. In Figure 9 the relations involved in crossings are genitive and adjective. In Figure 10 the relations are *kartā* and *karma*, which are the arguments of the verb, called *kāraka* relations in Sanskrit grammar. So in one sentence, there is a crossing between two *kāraka* relations. In another, the crossing is between non-*kāraka* relations. As we have seen earlier, the *kāraka* relations have mutual expectancies, while the non-*kāraka* relations such as genitive and adjective have unilateral expectancy. And in Figure 11, we see both types of crossings. Further we notice that while the graphs of Figure 9 and Figure 10 are well-nested, the graph of Figure 11 is ill-nested.

Now we describe the empirical observations of the dependency trees of BhG with special reference to the crossings involved and the well-nestedness.

4.1 Experiment

Sanskrit is a low resource language from the point of view of computational resources. For this experiment, we needed treebanks. A treebank developed under SHMT[3] consists of simple prose sentences, which hardly shows any crossings. There are some efforts to develop treebanks following the Universal Dependency (UD) (Hellwig et al., 2020). Since we aim at using the Pāṇinian grammar, the UD treebanks were not useful for our experiment. Therefore we decided to base our experiments on the same treebank that was used by Kulkarni et al. (2015). This treebank consists of verses from Śrīmad-Bhagavad-Gītā. It has 700 verses. Some verses were made up of more than one sentence while in some cases more than one verse formed one sentence. We followed the mīmāṁsaka's definition of a sentence given in section 3.

There were several ślokas which consisted of more than one sentence with an ellipsis of one or more word. For the evaluation purpose, we considered only complete sentences. So all the sentences with ellipsis of the verbs were not considered. For example, the first part of the verse BhG 1.15

pāñcajanyam hṛṣīkeśaḥ
deva-dattam dhanañjayaḥ |

consists of two sentences,

(a) *pāñcajanyam hṛṣīkeśaḥ (dadhmau)*
Pāñcajanya Hṛṣīkeśa (blew)

(b) *deva-dattam dhanañjayaḥ (dadhmau)*
Devadatta Dhanañjaya (blew)

There are two sentences, and both of them require a verb '*dadhmau*', which is to be borrowed from the next part. Such parts of verses which are devoid of a verb are not considered for the evaluation.

Similarly, in Sanskrit, the copula is absent. The tagging scheme demands the presence of a verb, and therefore, while tagging the verses, the copula is provided. Since in the original verses the copula is absent, we have not considered these verses/part of these verses where such copula is provided manually.

In order to decide whether the dependency graph is well-nested or not, we need to distinguish between the relations that show governance from those that do not show governance. All the relations that have mutual expectancy show governance. Table 1 lists all the relations that have mutual expectancy, and Table 2 shows all the relations that have only unilateral expectancy.

kartā	kartṛsamānādhikaraṇam
karma	karmasamānādhikaraṇam
karaṇam	sampradānam
apādānam	adhikaraṇam

Table 1: Relations with mutual expectancies

sambodhyaḥ	sambandhaḥ
śaṣṭhīsambandhaḥ	viśeṣaṇam
samuccitam	pratiṣedham
samuccayadyotakaḥ	nirdhāraṇam
prayojanam	hetuḥ
samānakālaḥ	pūrvakālaḥ
kriyāviśeṣaṇam	

Table 2: Relations with unilateral expectancies

In Figure 3, the edge $1 \rightarrow 3$, which crosses the edge $2 \rightarrow 5$, should be from Table 1. If either $1 \rightarrow 3$ is not from Table 1, or the two edges belong to two disjoint trees as in Figure 5, then the dependency graph is ill-nested. With the set of relations as described in Tables 1 and 2 we classified the dependency graphs of BhG verses. Table 3 shows the results of this empirical study.

[3]A project funded by Meity for the Development of Computational Tools and Sanskrit-Hindi Machine Translation.

Analysed sentences	1396	100.00%
Weakly non-projective	1153	82.59%
Only Well-nested	49	3.51%
Only Ill-nested	74	5.30%
Both Ill and well nested	120	8.60%

Table 3: Analysis of BhG

5 Discussions

The majority of the sentences (around 83%) have dependency graphs that are weakly non-projective. The remaining 17% graphs did not have planar graphs as they involved crossings of the dependency relations. Several of the sentences had more than one crossing. Some of these crossings show well-nestedness while the others show ill-nestedness. We notice that trees with only well-nested crossings are considerably less than trees with only ill-nested crossings. Further, there are almost double the number of sentences that have both ill-nested as well as well-nested crossings. Any graph, that involves both ill-nested as well as well-nested crossings, essentially is an ill-nested graph. Thus we notice that almost 14% of the sentences have ill-nested graphs. Thus every sixth sentence of the corpus has a non-planar graph, involving crossings between the disjoint graphs, with the majority of them being ill-nested. In order to understand more about these crossings, we looked at the relations involved in them. Table 4 shows the distribution of relations with mutual and unilateral expectancies in crossings.

We noted down the relations involved in crossings, and counted the number of instances of crossings that show well-nestedness or ill-nestedness. As expected, we noticed that, barring a few cases, at least one relation among the two relations involved in crossing has unilateral expectancy. Kulkarni et al. (2015) has discussed various examples of crossing where both the relations are with mutual expectancy.

Relations	Well-nested	Ill-nested
Mutual×Mutual	3	15
Mutual×Unilateral	109	136
Unilateral×Unilateral	82	99

Table 4: Relations involved in crossings

Now we provide one example each of the crossings with unilateral expectancies. The first one corresponds to a well-nested graph involving a crossing between a *kartā* and a *viśeṣaṇam*. This is from the first line of *śloka* 7.2.

jñānam te aham sa-vijñānam
idam vakṣyāmi a-śeṣataḥ |

(Eng: I will tell you this knowledge combined with realisation in detail.)

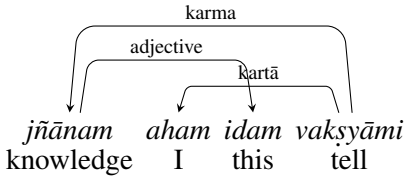


Figure 12: Analysis of BhG 7.2

In this tree, the two edges labelled adjective and *kartā* belong to two disjoint trees, and the head node '*vakṣyāmi*' of the *kartā* relation governs the head node '*jñānam*' of the adjectival relation. Hence this is a well-nested tree with a crossing between a relation of *kartā* having mutual expectancy with a relation of adjective having unilateral expectancy.

Now we present another example. This is 21st *śloka* from the same 7th chapter.

yaḥ yaḥ yāṁ yāṁ tanuṁ bhaktaḥ
śraddhayā arcitum icchati |

(Eng: Whichever form any devotee wants to worship.)

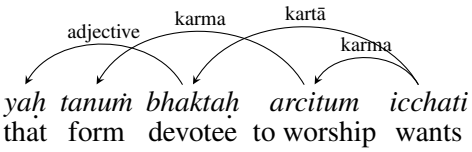


Figure 13: Analysis of BhG 7.21

In this dependency graph, we notice that there is a crossing between *karma* and an adjective, and neither of the heads governs the other, giving rise to an ill-nested graph. This graph also shows another crossing between a *kartā* and a *karma* relation, which corresponds to the well-nested graph.

Now we present two examples, where both the relations have unilateral expectancy. The first one is a well-nested graph which corresponds to the 4th *śloka* of 18th chapter.

niścayaṁ śruṇu me tatra
tyāge bharatasattama |

(Eng: O the most excellent among the descendants of Bharata, hear from me the firm conclusion regarding the abandonment.)

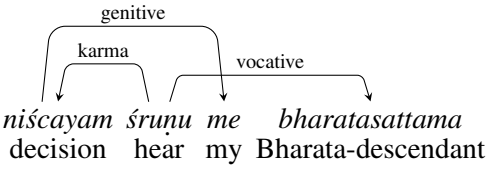


Figure 14: Analysis of BhG 18.4

In this graph, there is a crossing of two unilateral relations viz. genitive and vocative. The graph is well-nested, as the head of the genitive is governed by the head of the vocative relation.

The example of an ill-nested graph involving two unilateral relations is the first *śloka* of the 9th chapter.

idam tu te guhyatamam
pravakṣyāmi anasūyave |
jñānam vijñānasahitam
yat jñātvā mokṣyaseaśubhāt || BhG 9.1

(Eng: I shall now reveal to you the non-envious, the greatest secret, the knowledge combined with realisation, having known which you shall be free from evil.)

We show the partial graph with crossing relations.

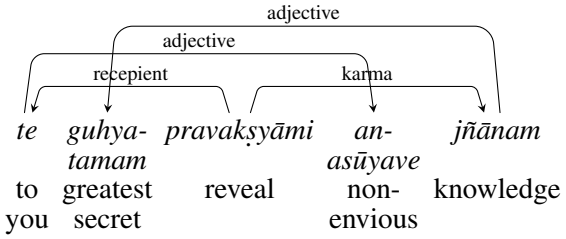


Figure 15: Analysis of BhG 9.1

In this graph, we see three crossings. The first one between the recipient and the adjective, the second one between the *karma* and the adjective, and the third one between the two adjectives. The first two crossings correspond to the well-nested graph. But the third one corresponds to the ill-nested one.

Finally, among the unilateral relations that contribute to either well-nested or ill-nested graphs, adjective, vocative, genitive and negation are prominent, followed by conjunction. Among the relations having mutual expectancy, *kartā*, *karma* and *adhikaraṇaṁ* are more prominent.

6 Conclusion

Sanskrit, as the common wisdom goes, is a free word order language. The Calder mobile model of Staal which conjunctures the free movement of the words within a phrase was found to be partially correct. Gillon through empirical study pointed out that there are certain cases of violation of this model. Later Kulkarni et al, again through the empirical study showed that the cases of violations of planarity correspond to the relations exhibiting unilateral expectancy. In this paper, we showed that there are as many cases of well-nested crossings as ill-nested ones. Thus not all syntactic structures of Sanskrit can be covered under the well-nested trees. A majority of non-planar graphs are ill-nested. In most of the cases, unilateral relations are involved in the violation of planarity as well as well-nestedness.

References

V. S. Apte. 1925. *The Student's Guide to Sanskrit Composition*, 9 edition. The Standard Publishing Company, Girgaon, Bombay.

R. N. Aralikatti. 1991. A note on word order in modern spoken Sanskrit and some positive constraints. In Hans Henrich Hock, editor, *Studies in Sanskrit Syntax*, pages 13–18. Motilal Banarsidass.

Hariprasad Bhagirath, editor. 1901. *Samāsacakra*. Jagadishwar Press, Mumbai.

Riyaz Ahmad Bhat and Dipti Misra Sharma. 2012. Non-projective structures in Indian language treebanks. In *Proceedings of the TLT11*, pages 25–30. Edições Colibri.

Manuel Bodirsky, Marco Kuhlmann, and Mathias Möhl. 2005. Well-nested drawings as models of syntactic structure. In *Proceedings of Tenth Conference on Formal Grammar and Ninth Meeting on Mathematics of Language*, pages 195–203.

Puneet Dwivedi and Easha Guha. 2017. Universal dependencies of Sanskrit. *International Journal of Advance Research, Ideas and Innovations in Technology*, 3(4).

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *16th International Conference on Computational Linguistics (COLING)*, pages 340–345.

Haim Gaifman. 1965. Dependency systems and phrase structure systems. *Information and Control*, 8:304–337.

Brendan S. Gillon. 1996. Word order in Classical Sanskrit. *Indian Linguistics*, 57(1):1–35.

Brendan S. Gillon. 2005. Subject predicate order in Classical Sanskrit. In Philip Scott, Claudia Casadio, and Robert Seely, editors, *Language and Grammar: Studies in Mathematical Linguistics and Natural Language*, pages 211–225. Center for the Study of Language and Information.

Oliver Hellwig, Salvatore Scarlata, Elia Ackermann, and Paul Widmer. 2020. The treebank of Vedic Sanskrit. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5137–5146, Marseille, France. European Language Resources Association.

Samar Husain and Shravan Vasishth. 2015. Non-projectivity and processing constraints: Insights from Hindi. In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*, pages 141–150, Uppsala, Sweden. Uppsala University.

Atmaram Narayan Jere. 2002. *Kārikāvalī*. Chowkamba Krishnadas Academy.

Marco Kuhlmann. 2010. *Dependency Structures and Lexicalized Grammars: An Algebraic Approach*. Springer-Verlang.

Marco Kuhlmann and Joakim Nivre. 2006. Mildly non-projective dependency structures. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 507–514. Association for Computational Linguistics.

Amba Kulkarni. 2019. *Sanskrit Parsing Based on the Theories of Śābdabodha*. D K Printworld.

Amba Kulkarni, Preeti Shukla, Pavankumar Satuluri, and Devanand Shukl. 2015. How free is 'free' word order in Sanskrit? In Peter Scharf, editor, *Sanskrit Syntax*, pages 269–304. Sanskrit Library.

Amba Kulkarni, Sanal Vikram, and Sriram K. 2019. Dependency parser for Sanskrit verses. In *Proceedings of the 6th International Sanskrit Computational Linguistics Symposium*, pages 14–27. Association for Computational Linguistics.

K Kunjunni Raja. 1963. *Indian Theories of Meaning*. Adayar Library and Research Center, Madras.

Prashanth Mannem, Himani Chaudhry, and Akshar Bharati. 2009. Insights into non-projectivity in Hindi. In *Proceedings of the ACL-IJCNLP 2009 Student Research Workshop*, pages 10–17. Association for Computational Linguistics.

Peter Scharf, Anuja Ajotikar, Sampada Savardekar, and Pawan Goyal. 2015. Distinctive features of poetic syntax: Preliminary results. In Peter Scharf, editor, *Sanskrit Syntax*, pages 305–324. Sanskrit Library.

J. Frits Staal. 1967. *Word Order in Sanskrit and Universal Grammar*, volume 5 of *Foundations of Language Supplementary Series*. D. Reidel Publishing Company, Dordrecht-Holland.

Himanshu Yadav, Ashwini Vaidya, and Samar Husain. 2017. Understanding constraints on non-projectivity using novel measures. In *Proceedings of the Fourth International Conference on Dependency Linguistics*, pages 276–286. Linköping University Electronic Press.

A Multi-modal Personality Prediction System

Chanchal Suman[1], Aditya Gupta[2], Sriparna Saha[1], and Pushpak Bhattacharyya[1]

[1]Department of Computer Science & Engineering, Indian Institute of Technology Patna, India
[2]Department of Electrical Engineering, Indian Institute of Technology Patna, India
email: {1821cs11, aditya.ee17, sriparna}@iitp.ac.in, pushpakbh@gmail.com

Abstract

Automatic prediction of personality traits has many real-life applications, e.g., in forensics, recommender systems, personalized services etc.. In this work, we have proposed a solution framework for solving the problem of predicting the personality traits of a user from videos. Ambient, facial and the audio features are extracted from the video of the user. These features are used for the final output prediction. The visual and audio modalities are combined in two different ways: averaging of predictions obtained from the individual modalities, and concatenation of features in multi-modal setting. The dataset released in Chalearn-16 is used for evaluating the performance of the system. Experimental results illustrate that it is possible to obtain better performance with a hand full of images, rather than using all the images present in the video.

1 Introduction

Our personality impacts a lot on our lives, affecting our life choices, mental health, well-being, and desires. Thus, automatic prediction of one's personality has many applications such as enhanced personal assistants, recommender system, job screening, forensics, psychological studies, etc. (Mehta et al., 2019). The big-five personality traits (Digman, 1990) are the most popular measures used in the literature. They are Extraversion, Neuroticism, Agreeableness, Conscientiousness, and Openness.

Most of the methods have utilized the CNN-based architectures for extracting the features from the images, mostly the facial features (Güçlütürk et al., 2016),(Gürpınar et al., 2016),(Biel et al., 2012). Mainly, researchers have used the late fusion strategy (averaging the predictions from all the modalities) for combining the results obtained from different modalities (audio, and video) (Güçlütürk et al., 2016, 2017; Gürpınar et al., 2016; Wei et al., 2017; Pianesi et al., 2008; André et al., 1999). There are very few works, which have employed early fusion strategy for developing the multimodal system (Yang et al., 2017; Kampman et al., 2018).

This has motivated us to develop a multimodal system which uses early fusion for combining the features generated from different modalities, and using the combined feature for final prediction. We have developed an audio-visual system, which extracts ambient features from video using ResNet (He et al., 2016), facial features using MTCNN (Zhang et al., 2016), and the audio features from the VGGish CNN (Hershey et al., 2017). Finally, those features are concatenated and then fed to the fully connected layer followed by a sigmoid layer for the final prediction. We have used the Chalearn-16 dataset for evaluating the performance of our system (Güçlütürk et al., 2016). An accuracy of 91.43% on the test data has been achieved using our proposed system.

The main contributions of this work are i) to the best of our knowledge, combined feature representation of images has been carried out for the first time for personality prediction. ii) The VGGish CNN has been used for extracting the audio features, and then those are used for the prediction. It has also been performed for the first time. From the results, it can be established that only some of the images extracted from different parts of the video are capable for successful training and testing of the model with good performance.

2 The Proposed Methodology

Our methodology consists of learning some features extracted from two different modalities, namely video and audio. They are discussed below:

2.1 Visual Modality

We have extracted two types of features from the visual modality, i) ambient, and ii) facial. Using

Proceedings of the 17th International Conference on Natural Language Processing, pages 317–322
Patna, India, December 18 - 21, 2020.

these two features, we have developed two different systems namely Amb-visual, and Fac-visual, respectively. Both of the architectures consist of three sub-parts, first is pre-processing step, second is the CNN architecture and the third is the final step to combine the features of the images to predict the big-five personality traits.

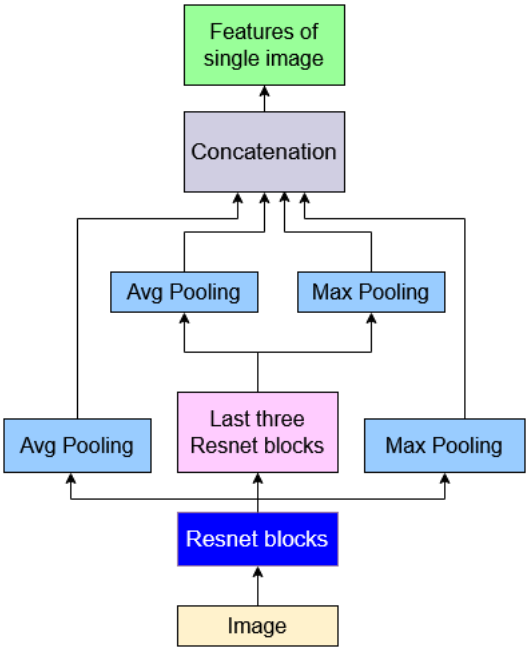


Figure 1: Extraction of Features from a Single Image using Deep Feature Extractor.

2.1.1 Preprocessing

The pre-processing of the data is important for extracting and learning appropriate features. The first step was to extract the images from the videos using FFmpeg tool in RGB color frame, and then resizing into 224x224 pixels.

i) Extraction of ambient features: we have extracted six equally spaced images to represent the whole video thus required RAM has decreased to around 20 Gb.

ii) Extraction of facial features: For the facial modality we first detect and align faces using Multi Task CNN (MTCNN) (Zhang et al., 2016). Again as before, only six images at equal intervals were considered from the set of images having faces.

2.1.2 Model Architecture

Here, we have discussed the model architectures for the Amb-visual, and the Fac-visual.

- We experimented with different CNN architectures and found that ResNet (He et al., 2016) with 101 layers worked best for extracting features from images. The feature vectors from last three layers are considered as the latent features of the image. For better and robust feature extraction instead of performing just max-pooling, we have performed both max and average pooling after the last convolution layer. Just like skip connection, these two feature vectors are concatenated to the upper feature vector which is extracted from the final ResNet block. This is named as deep feature extractor and it is depicted in figure 1.

- The model architecture for extracting features from the Fac-visual is same, as the Amb-visual. The only difference is the usage of MTCNN (Zhang et al., 2016) for extracting faces before passing it into deep feature extractor.

Apart from the feature extraction part, all the other components are same for the Amb-visual and the Fac-visual architecture.

2.1.3 Final classifier

The representations, learnt from the deep feature extractor module are fed to the final classification layer for the output prediction. We experimented in three settings, for analysing the behaviour of the system.

M1: In the first method, all six images were labeled with their corresponding video's numbers. The previously extracted features of each image are passed to a final fully connected layer with sigmoid as the activation function. This layer gives us the output values between 0, and 1 for the big five personality traits. The loss value achieved for each of the images is added to the final loss for training. Finally, the trait values for the video are obtained by considering the mean values of six images.

M2: In the second method, we concatenated the features of six images as a final feature vector, representing the video's visual feature. This feature vector is then passed to the fully connected and sigmoid layer for getting the final trait values of the video. In this method, loss of each image is not considered separately.

M3: For the third method, we try to use the fact that a video is a time-series data. By using an LSTM (Hochreiter and Schmidhuber, 1997), we wanted to learn more better features so that image at time 't' could be represented using information from previous time steps as well. For that,

we passed the extracted features into an LSTM of appropriate hidden dimension with several layers. After that, the outputs of LSTM for different time-steps are collected, and then concatenated for getting the final feature vector of a video. This feature vector is then passed to a fully connected + sigmoid layer to extract trait values for a video.

2.2 Audio Modality

For the audio modality, the VGGish CNN (Hershey et al., 2017), along with below mentioned pre-processing steps are used. The pre-processing and architecture are explained in the following section.

2.2.1 Preprocessing

Firstly, all audios are re-sampled to 16 kHz mono. A spectrogram is computed using magnitudes of the Short-Time Fourier Transform with a window size of 25 ms, a window hop of 10 ms, and a periodic Hann window. A mel spectrogram is computed by mapping the above spectrogram to 64 mel bins covering the range 125-7500 Hz. Then log of mel spectrogram is computed with a small offset of 0.01 to stabilize mel spectrogram values. These features are then framed into non-overlapping examples of 0.96 second, where each example covers 64 mel bands and 96 frames of 10 ms each.

2.2.2 Model Architecture

After the pre-processing, a 2d feature array of shape 96x64 for each 1 second was obtained. Thus, for a 15 second video there are 15 such feature vectors. This feature vector is then passed to the VGGish CNN architecture, that has many 2d convolution layers. This CNN outputs a 128 length embedding for each second, that was further used to train a classifier for getting traits. Like before, we initialised the weights of our convolution filters with the pre-trained weights of VGGish CNN trained on large Youtube dataset for warm start of training.

After getting audio features, for each second of the video we have fifteen 128 length vectors, two methods were experimented for further combining the features for regression.

Audio-M1: In the first method, these features are passed into few layers of LSTMs of appropriate hidden layers and then the outputs of LSTM are concatenated to get a final feature vector of whole audio. Then a fully connected layer with sigmoid activation function is applied.

Audio-M2: In the second method, instead of using LSTMs, we simply concatenate the audio features and then apply a fully connected layer and sigmoid. This network is called deep audio feature extractor.

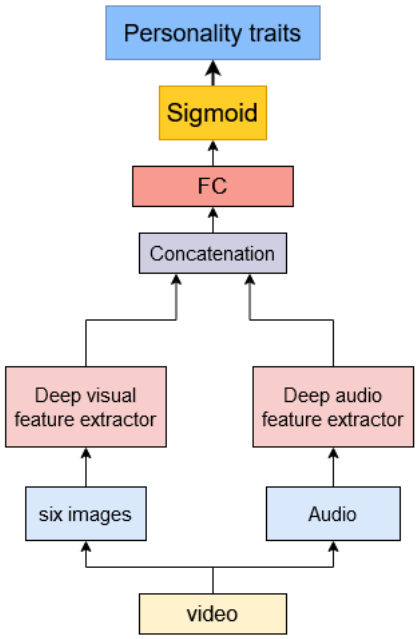


Figure 2: The Proposed Multi-modal System

2.3 Combining Modalities

We experimented with two approaches for combining the modalities. They are: 1) Average: we simply took the average of predicted trait values of different modalities. ii) Concat: we concatenated the obtained features of different modalities and then applied fully connected layer with sigmoid. It is depicted in Figure 2.

Table 1: Performance of the Proposed Model on Validation Data

Modality	Accuracy (in %)
Audio-M1	90.29
Audio-M2	90.64
Amb-visual (M1)	91.27
Amb-visual (M2)	91.19
Amb-visual (M3)	90.65
Fac-visual (M1)	90.90
Fac-visual (M3)	90.51
Amb-visual+audio (Concat)	91.44
Amb-visual+audio (Average)	91.56
Amb-visual+Fac-visual+audio (Average)	91.62

Table 2: Class-wise accuracy (in %) values of the proposed model on test data

	Average	Extraversion	Agreeableness	Conscientiousness	Neuroticism	Openness
Ours (Fusion(average))	91.43	91.53	91.29	92.06	91.18	91.07
PersEmoN (Zhang et al., 2019)	91.7	92.0	91.4	92.1	91.4	91.5
NJU-LAMDA (Wei et al., 2017)	91.3	91.3	91.3	91.7	91.0	91.2
evolgen (Subramaniam et al., 2016)	91.2	91.5	91.2	91.2	91.0	91.2
DCC(Güçlütürk et al., 2016)	91.1	91.1	91.0	91.4	90.9	91.1
ucas (**?**)	91.0	91.3	90.9	91.1	90.6	91.0

3 Results and Comparison

In this section, we have discussed the results obtained for our proposed approach. We have compared these results with the existing works too.

3.1 Results and Discussion

The ECCV ChaLearn LAP 2016 data, is used for experimentation, having $10,000$ videos. Out Of the $10,000$ videos, $6,000$ videos are used for training phase and $2,000$ videos for both validation and testing (Wei et al., 2017). The accuracy is used as the performance measure, for evaluating the proposed system.

In all the modalities, models without LSTM layers performed better than the ones with them as shown in table 1. Validation accuracies with LSTM are 90.65, 90.51, 90.29 for Amb-visual, Fac-visual, and the Audio modality, respectively, whereas the corresponding accuracies without LSTM are 91.27 , 90.90 and 90.64, respectively. In the Amb-visual, and the Fac-visual, the highest validation accuracies were achieved by using average and max pooling on ultimate and penultimate ResNet layers and concatenating these features of six images (M2). For M1 (averaging the prediction of all the six images), we achieved an accuracy of 91.19 for Amb-visual. We didn't evaluate the M1 for Fac-visual, as the accuracy of M1 for Amb-visual is lesser than M2.

Similar to the Visual modality, the method based on LSTM layers (Audio-M1), proved disadvantageous as validation accuracy with LSTM layers is 90.29 and without LSTM (Audio-M2) is 90.64.

Table 3: Comparison with Other Works

Modality	**Our Method**	NJU-LAMDA (Wei et al., 2017)	PersEmoN (Zhang et al., 2019)
Visual	91.13	91.16	**91.7**
Audio	**90.16**	89.50	–
Video+ Audio (Avg)	**91.43**	91.30	–

Since, audio is a time series data, LSTMs should have increased the accuracy. But this is not observed in the obtained results. It proves that LSTM was futile and has only led to overfitting.

We have applied two different approaches for combining the two modalities, i) Average, and ii) Concat. By averaging the two best performing modalities (Amb-visual, and audio), a validation accuracy of 91.56% is attained. Concatenation of the features generated from different modalities (Amb-visual, audio) resulted in a validation accuracy of 91.44%. After that, we calculated the performance using averaging of predictions of all the three modalities, and achieved an accuracy of 91.62%. Average accuracies of 91.13%, 90.16%, and 91.43% are achieved using the best models for video(Amb-visual), audio(Audio-M2), and the fusion (Average), respectively on test data.

3.2 Comparison with Other Works

The best performing system on the Chalearn-16 dataset is developed by the (Zhang et al., 2019).

Emotion and personality, both features are fused together in (Zhang et al., 2019), for analysing the effects of emotion on personality prediction. The methodology developed by (Wei et al., 2017) is the second best performing work.

We tried two different approaches for combining the visual and the audio features, averaging and the concatenation. The averaging of the predictions generated by three modalities has attained an accuracy of 91.43%. The detailed class-wise accuracy for each of the class and the comparative results are shown in table 2, and 3 respectively. It can be seen that, our proposed approach (video+audio(average)) attains better performance than the method proposed in (Wei et al., 2017). We have achieved an accuracy of 91.43% for the visual modality, while 91.30% is reported by (Wei et al., 2017). This shows that, only handful of images extracted from different parts of the video are enough for successful training and testing with good performance. The researchers in (Wei et al., 2017), used 100 images for making the visual system, while we have extracted only 6 images. For the audio modality, an increment of 1.14% is attained with respect to the existing one. But our developed system, could not outperform the performance of the multi-task based methodology. The reason can be, the incorporation of emotion features in their model.

From the obtained results, it can be concluded that multi-modality helps as the concatenation of features extracted from visual and audio improves the accuracy in comparison to the single one.

4 Conclusion and Future Work

Personality prediction reveals the overall characteristics of a user. In this work, we have proposed a deep multi-modal system for personality prediction given a video. It extracts features from a video, and those are then used in the neural network setting for the final prediction. From the obtained experimental results, we can conclude the following : i) only handful of images from different parts of video are enough for successful training and testing with good performance, ii) averaging the predictions of different modalities yields better performance than the simple concatenation of the modalities in the multi-modal setting.

We are planning to improve the fusion strategy for combining the different modalities. We will try to use emotion features and different types of attention mechanisms like weighted attention, self-attention etc. for combining the modalities.

Acknowledgments

Sriparna Saha would like to acknowledge the support of SERB WOMEN IN EXCELLENCE AWARD 2018 for conducting this research. This research is also supported by Ministry of Electronics and Information Technology, Government of India.

References

Elisabeth André, Martin Klesen, Patrick Gebhard, Steve Allen, and Thomas Rist. 1999. Integrating models of personality and emotions into lifelike characters. In *International Workshop on Affective Interactions*, pages 150–165. Springer.

Joan-Isaac Biel, Lucía Teijeiro-Mosquera, and Daniel Gatica-Perez. 2012. Facetube: predicting personality from facial expressions of emotion in online conversational video. In *Proceedings of the 14th ACM international conference on Multimodal interaction*, pages 53–56.

John M Digman. 1990. Personality structure: Emergence of the five-factor model. *Annual review of psychology*, 41(1):417–440.

Yağmur Güçlütürk, Umut Güçlü, Xavier Baro, Hugo Jair Escalante, Isabelle Guyon, Sergio Escalera, Marcel AJ Van Gerven, and Rob Van Lier. 2017. Multimodal first impression analysis with deep residual networks. *IEEE Transactions on Affective Computing*, 9(3):316–329.

Yağmur Güçlütürk, Umut Güçlü, Marcel AJ van Gerven, and Rob van Lier. 2016. Deep impression: Audiovisual deep residual networks for multimodal apparent personality trait recognition. In *European Conference on Computer Vision*, pages 349–358. Springer.

Furkan Gürpınar, Heysem Kaya, and Albert Ali Salah. 2016. Combining deep facial and ambient features for first impression estimation. In *European Conference on Computer Vision*, pages 372–385. Springer.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. 2017. Cnn architectures for large-scale audio classification. In *2017 ieee international conference on acoustics, speech and signal processing (icassp)*, pages 131–135. IEEE.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Onno Kampman, Elham J Barezi, Dario Bertero, and Pascale Fung. 2018. Investigating audio, video, and text fusion methods for end-to-end automatic personality prediction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 606–611.

Yash Mehta, Navonil Majumder, Alexander Gelbukh, and Erik Cambria. 2019. Recent trends in deep learning based personality detection. *Artificial Intelligence Review*, pages 1–27.

Fabio Pianesi, Nadia Mana, Alessandro Cappelletti, Bruno Lepri, and Massimo Zancanaro. 2008. Multimodal recognition of personality traits in social interactions. In *Proceedings of the 10th international conference on Multimodal interfaces*, pages 53–60.

Arulkumar Subramaniam, Vismay Patel, Ashish Mishra, Prashanth Balasubramanian, and Anurag Mittal. 2016. Bi-modal first impressions recognition using temporally ordered deep audio and stochastic visual features. In *European Conference on Computer Vision*, pages 337–348. Springer.

Xiu-Shen Wei, Chen-Lin Zhang, Hao Zhang, and Jianxin Wu. 2017. Deep bimodal regression of apparent personality traits from short video sequences. *IEEE Transactions on Affective Computing*, 9(3):303–315.

Karen Yang, S Mall, and N Glaser. 2017. Prediction of personality first impressions with deep bimodal lstm.

Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. 2016. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503.

Le Zhang, Songyou Peng, and Stefan Winkler. 2019. Persemon: A deep network for joint analysis of apparent personality, emotion and their relationship. *IEEE Transactions on Affective Computing*.

D-Coref: A Fast and Lightweight Coreference Resolution Model using DistilBERT

Chanchal Suman[1], Jeetu Kumar[2], Sriparna Saha[1], and Pushpak Bhattacharyya[1]

[1]Department of Computer Science & Engineering, Indian Institute of Technology Patna, India
email: {1821cs11, sriparna}@iitp.ac.in, pushpakbh@gmail.com
[2]Department of Computer Science, RKMVERI, Belur Math, Howrah, India
email: 0jkpandey@gmail.com

Abstract

Smart devices are often deployed in some edge-devices, which require quality solutions in limited amount of memory usage. In most of the user-interaction based smart devices, coreference resolution is often required. Keeping this in view, we have developed a fast and lightweight coreference resolution model which meets the minimum memory requirement and converges faster. In order to generate the embeddings for solving the task of coreference resolution, DistilBERT, a light weight BERT module is utilized. DistilBERT consumes less memory (only 60% of memory in comparison to BERT-based heavy model) and it is suitable for deployment in edge devices. DistilBERT embedding helps in 60% faster convergence with an accuracy compromise of 2.59%, and 6.49% with respect to its base model and current state-of-the-art, respectively.

1 Introduction

Edge devices require natural language processing (NLP) for understanding users' input[1]. Whenever it comes to user interaction, coreference resolution becomes an important task for analysing user's input. It involves determining all referring expressions that point to the same real-world entity. A grouping of referring expressions with the same referent is called a coreference chain or cluster. The goal of a coreference resolution system is to output all the coreference chains of a given text (Martschat and Strube, 2015; Ferreira Cruz et al., 2020).

Several works on coreference resolution are available in the literature having very high accuracy (Lee et al., 2017, 2018; Kantor and Globerson, 2019; Joshi et al., 2019; Fei et al., 2019). These models use ELMo (Lee et al., 2018) and BERT (Joshi et al., 2019) for learning the semantic space of the input. Because of the use of such heavy transformers with millions of parameters, these models require a lot of memory. However, smart devices like smartphones should be responsive, light-weight, and energy-efficient models. This motivates us to design a light-weighted coreference resolution model which is suitable for the deployment in smart-devices.

We contributed in word context representation of c2f-model (Lee et al., 2018), by forming it from embedding generated by DistilBERT instead of ELMo. We use c2f-model as our baseline model, since this is used as base model in all the recent works ((Kantor and Globerson, 2019), (Joshi et al., 2019)). DistilBERT is a smaller, faster, cheaper, and light-weight distilled version of BERT, which is approx. 97% efficient in comparison to BERT. It is 40% smaller in size, and 60% faster (Sanh et al., 2019). The embeddings are generated from the DistilBERT for learning the semantic space of the sentences. After the generation of embeddings, they are passed to the bidirectional LSTM, followed by span head for calculation of mention scores. These mention scores are used for forming the coreference chain using hierarchical clustering as defined in (Lee et al., 2018).

The standard CoNLL-2012 (Pradhan et al., 2012) dataset is utilized for the performance evaluation of our proposed model. Experimental results show that, the developed system requires only 60% memory for execution, in comparison to the BERT-based heavy models, while remaining 97% efficient, and 60% faster converging too. We have also shown that 768 embedding dimension is sufficient for word context embedding generation from DistilBERT.

[1]https://www.iotforall.com/iot-natural-language-processing/

Proceedings of the 17th International Conference on Natural Language Processing, pages 323–328
Patna, India, December 18 - 21, 2020.

2 The Proposed Approach

In order to utilize the embeddings generated by DistilBERT for extracting the coreference chains, we have integrated the recently proposed higher-order coreference model proposed in (Lee et al., 2018) in our system. We refer to this work as c2f-model.

2.1 Overview of c2f-model

For each mention span u, the model learns a distribution $P(\cdot)$ over possible antecedent spans v, as shown in equation 1. The scoring function s(u,v) between spans u and v takes g_u and g_v as its inputs. It uses fixed-length span representations. The scoring function consists of a concatenation of three vectors: the LSTM states of both the span endpoints and an attention vector computed over those span tokens. The score $s(u, v)$ is computed by the mention score of u ($s_m(u)$), mention score of v ($s_m(v)$), the joint compatibility score ($s_c(u, v)$) of u and v. The mention score of a span signifies the probability of a span to be a mention. The joint compatibility score signifies the probability of the two spans as corefering. The components are computed as follows:

$$P(v) = \frac{e^{s(u,v)}}{\sum_{v' \in V} e^{s(u,v')}} \quad (1)$$

$$s(u, v) = s_m(u) + s_m(v) + s_c(u, v) \quad (2)$$

$$s_m(u) = FFNN_m(g_u) \quad (3)$$

$$s_c(u, v) = FFNN_c(g_u, g_v, \phi(u, v)) \quad (4)$$

where $FFNN(\cdot)$ represents a feed forward neural network and $\phi(u, v)$ represents speaker and meta-data features. Antecedent distribution is used for further refinement of these generated span representations. Finally coreference chain is formed using the scores generated from the softmax layer.

2.2 Extraction of Embedding from DistilBERT for word context representation

Extraction of embeddings from ELMO is shown in (Peters et al., 2018). We have shown the embedding extraction from DistilBERT for word representation in Fig. 1. This extraction of word representation is performed in 4 steps, which are explained below.

Conversion of Term-Tokens into WordPiece tokens: DistilBERT takes token embedding and position embeddings as input (Sanh et al., 2019). Thus, complete sentences are formed from the Term-Tokens, and passed to the DistilBERT tokenizer. DistilBERT takes WordPiece tokens generated by the tokenizer and merges the initial embeddings and the position embeddings as the final input for it.

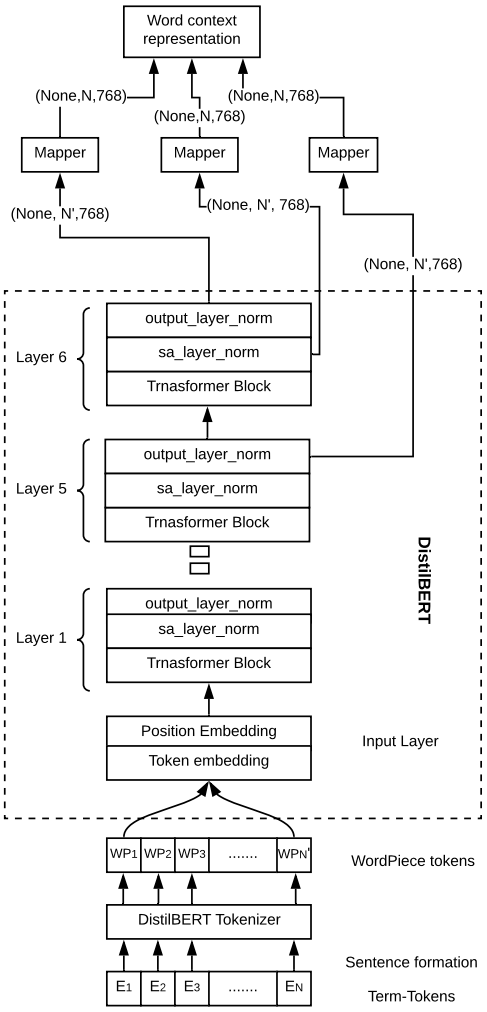


Figure 1: Word context representation from DistilBERT

Collecting outputs from DistilBERT: After, the generation of WordPiece tokens, these are given as input to the DistilBERT for the generation of word embeddings. For generating the word context representations from the ELMo, the three features 1) output of left-LSTM, 2) output of right-LSTM and 3) final embedding have been considered in the c2f-model. Similarly, to strengthen the learning from word context representation we generate word context representation from triplet of embedding outputs. We consider the raw form of embedding

outputs from sa_layer_norm of Layer-6 and output_layer_norm of Layer-5 and final embedding output from the output_layer_norm of Layer-6 of DistilBERT. Word context representation means representation of word in the input sentence. *Word representation* as defined in c2f-model, are generated by character embedding using GloVe(Pennington et al., 2014) vector.

Mapping Embedding from WordPiece tokens to Term-tokens : The dimension of embedding matrix generated from DistilBERT is $(None, N', 786)$. Here, N' is the maximum of the number of WordPice tokens for a sample point in the batch. Learning the coreference in context of WordPiece token is complex to understand, and its analysis and explanation seem unusual. So we have mapped the output of WordPiece token to Term-Token by averaging the corresponding WordPiece embeddings.

Let, in a batch of size B, N be the maximum of number of Term-Tokens in a sample, $N^{'}$ be the maximum number of WordPiece tokens, and $i, j, k, l \in \mathbf{N}$. Let, WordPiece token generated by DistilBERT tokenizer be $WPT = \langle WP_1, WP_2, WP_3, \ldots, WP_{N'} \rangle$ (when the number of tokens in WordPiece token is less than $N^{'}$, then post-padding is done to get it) for input Term-Token, $ET = \langle E_1, E_2, E_3, \ldots, E_N \rangle$. Let, the embedding output generated from DistilBERT be $EmbOut^{'}$, which is a matrix of order $(B, N^{'}, 768)$, where

$$EmbOut^{'}[i] = \left[e^{'}_{j,k}\right]_{1 \leq j \leq N';\ 1 \leq k \leq 768};$$

$\forall 1 \leq i \leq B$ Then, we map it to the $EmbOut$ matrix of order $(B, N, 768)$ *i.e.*,

$$EmbOut[i] = [e_{j,k}]_{1 \leq j \leq N;\ 1 \leq k \leq 768};$$

$\forall 1 \leq i \leq B$ where

$$e_{j,k} = e^{'}_{j,k}; \tag{5}$$

$$\text{if } WP_j = E_j \times \frac{1}{l} \sum_{p=1}^{l} e^{'}_{j+p,k}; \tag{6}$$

$$\&\text{if } \Psi(j,l) = True \tag{7}$$

$\forall\ 1 \leq k \leq 768$ and the function $\Psi(j,l)$ *returns True* if the WordPiece tokens $\langle WP_{j+1}, \ldots, WP_{j+l} \rangle$ lead to term-token, E_j. Similar procedure is followed to get the embedding output from the rest of the two layers.

Formation of the final word context representation: The output from Layer-6, sa_layer_norm of Layer-6, and output of Layer_5 are separately passed to mapper and mapped output $m1, m2, and, m3$ are collected. Finally, $m1, m2, and, m3$ are concatenated for generation of the word context representation. This mapping also reduces the second dimension of word context representation from N' to N. Thus, the order of word context representation becomes $(None, N, 2304)$, where N is the maximum number of tokens in a sample in the batch; this reduction makes the model to work with less space too.

2.3 Overview of the proposed system

The word and character embeddings are generated via DistilBERT and Glove, respectively. The word embedding generation through DistilBERT is discussed in the subsection 2.2. Character embeddings are generated through Glove similar to the c2f-model. The flow of our model after embedding generation is same as that of the c2f-model. The embeddings are fed to bidirectional LSTM to learn encoded representations for the words. The encoded features are further passed ahead to form the span head and span representation with span head feature. These span representations are then used for calculating the coreference score. Mention score and antecedant score are used for calculating the final coreference score. The formula for these calculations is shown in the Equation 1. For determining the final probability distribution between different spans, softmax is applied. At last, hierarchical clustering is used to form the coreference chain using the generated probability distribution.

3 Dataset used and experimental set-up

CoNLL-2012 shared task corpus is a standard coreference resolution corpus (Pradhan et al., 2012). We have used the English-based corpus for evaluating the performance of our proposed approach.

Our experimental setup is almost similar to that of c2f-model and we have modified some parts of their code to generate word context representation from DistilBERT embeddings, which are:

1) The ELMo embeddings are replaced by the DistilBERT embeddings which are lighter and faster.

2) We have experimented with *word context representation*, generated from DistilBERT. The two different experimental setups are discussed below:

i) D-Coref-Small: In our proposed D-coref model, we have extracted the embeddings from the three layers of DistilBERT for generating the word con-

text representation. The order of generated embedding is $(None, N, 768)$ and the order of word context representation is $(None, N, 2304)$.

ii) D-Coref-Large: For higher dimensional word context representation, we have extracted the embedding outputs from layer-4 of DistilBERT for raw embedding representation in addition with embeddings of D-coref-Small. Thus, the order of word context representation for this setup is $(None, N, 3072)$ similar to c2f-model.

Table 1: Comparison with previous works

	MUC	B^3	CEAF	Avg F1
(Lee et al., 2017)	75.8	65.0	60.8	67.2
(Lee et al., 2018)	80.4	70.8	67.6	73.0
(Joshi et al., 2019)	83.5	75.3	71.9	76.9
D-coref-Large	78.15	67.94	64.76	**70.28**
D-coref-Small	78.27	68.09	64.87	**70.41**

4 Results and Analysis

In this section, we have discussed the performance of our model on the standard CoNLL-2012 dataset, along with different features of the model.

4.1 Performance Evaluation

We have reported precision, recall and F1-scores of the B^3, MUC, and,CEAF metrics, and average F1 score (main evaluation metric) of all these three metrics as per the previous papers (Pradhan et al., 2012). The results obtained from our proposed approach is tabulated in table 1, and the detailed comparison is shown in table 2. Our baseline is the c2f-model with ELMo input features, which achieves an average F1 of 73.0%. We have achieved an average F1 of 70.41% for D-Coref-Small, and 70.28% for D-Coref-Large. Our experiments show that getting word context representation in the dimension of $(None, N, 2304)$ is sufficient. After observing the performance and the size of the model, we consider the D-Coref-small as our final model. The performance of D-Coref-small is 6.49% less than the current state-of-the-art (Joshi et al., 2019) and 2.59% less than the c2f-model. This performance matches with the claim of 3% less language understanding capability of the DistilBERT model[2]. We have a loss of approx 6% in performance, but this is the inherent nature of DistilBERT. At the same time, our model has become faster and lightweight due to the usage of the faster and lighter DistilBERT.

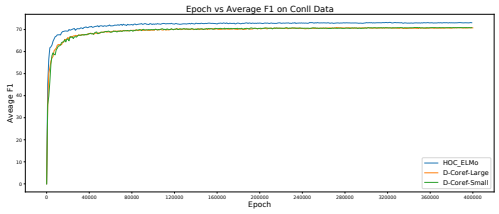


Figure 2: Epoch versus Average F1 curve

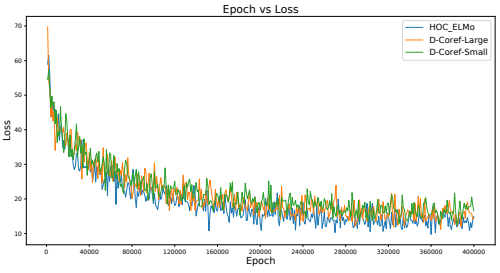


Figure 3: Epoch versus Loss curve

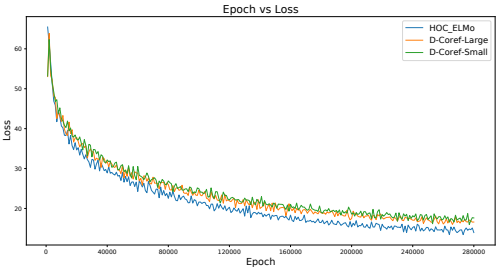


Figure 4: Epoch versus Average loss graph

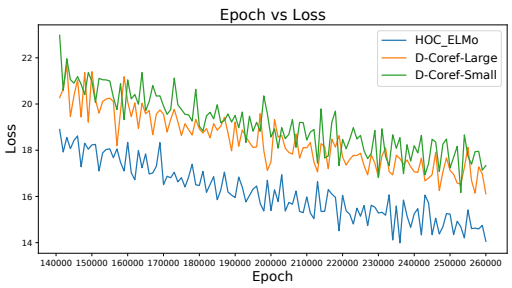


Figure 5: Epoch versus Average Loss sub graph

The detailed comparison table, with all the performance metrics are shown in Table 2. The epoch vs loss and epoch vs average F1 curves are shown in Figures 2, 3, 4, and 5.

[2]https://medium.com/huggingface/distilbert-8cf3380435b5

Table 2: Comparison with previous works

	MUC			B^3			CEAF			
	P	R	F1	P	R	F1	P	R	F1	Avg F1
(Martschat and Strube, 2015)	76.7	68.1	72.2	66.1	54.2	59.6	59.5	52.3	55.7	62.5
(Clark and Manning, 2015)	76.1	69.4	72.6	65.6	56.0	60.4	59.4	53.0	56.0	63.0
(Wiseman et al., 2015)	76.2	69.3	72.6	66.2	55.8	60.5	59.4	54.9	57.1	63.4
(Wiseman et al., 2016)	77.5	69.8	73.4	66.8	57.0	61.5	62.1	53.9	57.7	64.2
(Clark and Manning, 2016)	79.2	70.4	74.6	69.9	58.0	63.4	63.5	55.5	59.2	65.7
(Lee et al., 2017)	78.4	73.4	75.8	68.6	61.8	65.0	62.7	59.0	60.8	67.2
(Lee et al., 2018)	81.4	79.5	80.4	72.2	69.5	70.8	68.2	67.1	67.6	73.0
(Joshi et al., 2019)	84.7	82.4	83.5	76.5	74.0	75.3	74.1	69.8	71.9	76.9
D-coref-Large	80.45	75.97	78.15	71.32	64.86	67.94	66.59	63.03	64.76	**70.28**
D-coref-Small	80.85	75.86	78.27	71.91	64.65	68.09	62.69	67.2	64.87	**70.41**

4.2 Characteristics df the Proposed Model

Our proposed model is fast and light-weight. Here, we have discussed these two properties in detail.

Fast: From the epoch vs loss graph (fig. 3), we observed that model does not show any improvement after 240K. But after examining the average F1 plot (fig. 2), it is evident that the model has converged at 200K and there is no improvement in average F1 after 200K, while the c2f-model had converged at 400K epochs. In this way, the developed model is 60% faster, this behaviour also matches with the claim of faster learning capability of the DistilBERT (Sanh et al., 2019).

Light-weight: DistilBERT is a very light-weight model, with 66 millions of parameters, while the transformer ELMo has 465 millions of parameters (Sanh et al., 2019). Thus it can meet the memory requirements of edge devices. The requirement of fewer parameters for DistilBERT is the main motivation of this work. At the same time, the reduced word context representation dimension of D-coref-small has also lowered the model size, because the entire learning dimension depends on word context representation as it flows throughout the model.

In the view of these advantages, it is evident that our model is suitable for small devices with some compromise in performance. The size of the DistilBERT is reduced by 40% in comparison to BERT model, while it retains 97% of the language understanding capabilities of BERT and is 60% faster (Sanh et al., 2019). Thus, the usage of DistilBERT embeddings makes our model faster and lighter.

5 Conclusion and Future Work

We have devised a fast and light-weight coreference resolution model using DistilBERT. In order to generate a faster and light-weight model, the accuracy gets compromised. Word context representation in reasonable lower dimension can work like representation in higher dimension with proper tuning. Our developed system requires only 60% memory for execution, in comparison to the BERT-based heavy models, while remaining 97% efficient too. Thus, it is suitable for edge devices. In future we will try to come up with a model having better performance with same or lesser space requirement.

Acknowledgments

Dr. Sriparna Saha gratefully acknowledges the Young Faculty Research Fellowship (YFRF) Award, supported by Visvesvaraya Ph.D. Scheme for Electronics and IT, Ministry of Electronics and Information Technology (MeitY), Government of India, being implemented by Digital India Corporation (formerly Media Lab Asia) for carrying out this research.

References

Kevin Clark and Christopher D Manning. 2015. Entity-centric coreference resolution with model stacking. 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 1405–1415.

Kevin Clark and Christopher D Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. *arXiv preprint arXiv:1609.08667*.

Hongliang Fei, Xu Li, Dingcheng Li, and Ping Li. 2019. End-to-end deep reinforcement learning based coreference resolution. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 660–665.

André Ferreira Cruz, Gil Rocha, and Henrique Lopes Cardoso. 2020. Coreference resolution: Toward end-to-end and cross-lingual systems. *Information*, 11(2):74.

Mandar Joshi, Omer Levy, Daniel S Weld, and Luke Zettlemoyer. 2019. Bert for coreference resolution: Baselines and analysis. *arXiv preprint arXiv:1908.09091*.

Ben Kantor and Amir Globerson. 2019. Coreference resolution with entity equalization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 673–677.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. *arXiv preprint arXiv:1707.07045*.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. *arXiv preprint arXiv:1804.05392*.

Sebastian Martschat and Michael Strube. 2015. Latent structures for coreference resolution. *Transactions of the Association for Computational Linguistics*, 3:405–418.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 1–40. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Sam Wiseman, Alexander M Rush, and Stuart M Shieber. 2016. Learning global features for coreference resolution. *arXiv preprint arXiv:1604.03035*.

Sam Joshua Wiseman, Alexander Matthew Rush, Stuart Merrill Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. 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)*. Association for Computational Linguistics.

Semantic Slot Prediction on low corpus data using finite user defined list

Bharatram Natarajan, Simma Dharani, Chirag Singh, Anish Nediyanchath and **Sreoshi Sengupta**
research.samsung.com
{bharatram.n, simma.d, c.singh, anish.n, s.sengupta}@samsung.com

Abstract

Semantic slot prediction is one of the important task for natural language understanding (NLU). They depend on the quality and quantity of the human crafted training data, which affects model generalization. With the advent of voice assistants exposing AI platforms to third party developers, training data quality and quantity matters for any machine learning algorithm to learn and generalize properly. AI platforms provides provision to add custom external plist defined by the developers for the training data. Hence we are exploring dataset, called LowCorpusSlotData, containing low corpus training data with larger number of slots and significant test data. We also use external plist for the above dataset to aid in slot identification. We experimented using state of the art architectures like Bi-directional Encoder Representations from Transformers (BERT) with variants and Bi-directional Encoder with Custom Decoder. To address the low corpus problem, we propose a pipeline approach where we extract candidate slot information using the external plist extractor module and feed as input along with utterance.

1 Introduction

In recent market, many voice assistants like Samsung Bixby, Amazon Alexa are providing AI platforms for developers outside to develop their custom applications. In these platforms, developers add the data for their custom made applications which is being used by the voice assistant for the training its NLU components. The main responsibility of NLU is to understand the user utterance and to determine the domain, intent and slots. Domain Prediction extracts the domain in which the utterance belongs to i.e., the application that will execute the user intention. Intent Detector is to extract the user intention or the action that we will execute inside the application. Slot Filler is to extract the semantic slots or objects of interest on which we execute the action. Example is name entity finding like name of person, location and date time. The data, developed by the developers, might be small for training slot-detection task when compared to intent detection or domain prediction task. Therefore, the current scope of this paper is limited solving the Slot Filler task. To explain Slot Filler task with an example, consider the utterance "Is the Barbeque Nation closed at this time?". Slot Filler task is to identify "Barbeque Nation" as BusinessName and "Closed" as OpenHourDescriptor from the utterance.

Lots of work is going on the field of slot prediction as independent task. Shin et al. (2018) proposes the architecture based on encoder-decoder attention model with aligned input where Bi-GRU as encoder and GRU decoder which learns jointly both slot filling and delexicalized sentence generation. Saha et al. (2018) proposes the variants of LSTM and GRU integrated with the CRF layer at end for the task of slot filling. Mesnil et al. (2014) recommends the usage of RNN for the slot prediction. Jaech et al. (2016) performs an experiment using a multi-task model with open vocabulary embeddings increases the generalizability by which the data required for the training the slot-filling is minimalised. Huang et al. (2015) proposes a various LSTM-CRF models for sentence tagging. Kurata et al. (2016) proposes the encoder-labeler LSTM which performs slot filling conditioned on the encoded sentence-level information which was generated by LSTM. Shi et al. (2016) recommends a recurrent support vector machine, which is a combination of recurrent neural network, and a structured support vector machine for the slot tagging. Liu et al. (2020) proposes a cross-domain slot filling using Bi-LSTM for handling the limitation of data and unseen slot types. Zhang et al. (2018) proposes a capsule based neural network model,

Proceedings of the 17th International Conference on Natural Language Processing, pages 329–333
Patna, India, December 18 - 21, 2020.

which accomplishes slot filling and intent detection via a dynamic routing-by-agreement schema. Firdaus et al. (2019) recommends a multi-task hierarchical approach using the CNN, RNN to get the contextual information and uses CRF for model label dependency. All the above approaches suffers from the lack of intent information to enhance slot task.

To circumvent the above limitation, exploration on slot prediction as joint task with intent gained momentum. Wang et al. (2020) proposes a new architecture for joint intent detection and slot filling based on pre-trained BERT (Chen et al., 2019), added the self-attention and slot gate with CRF which improvement in slot filling. Gangadharaiah and Narayanaswamy (2019) proposes attention information (calculating the attention of current encoder with respect to previous encoders), in addition to encoder during each decoder step for predicting joint intent and slot.

Inspired by performance of the above state of the art architectures in slot prediction work, we are exploring architectures namely BERT (Chen et al., 2019) and Encoder with Custom Decoder (Gangadharaiah and Narayanaswamy, 2019) on open source dataset with external list information.

We organize rest of the paper as follows. Section 2 describes the Proposed Approach. Section 3 describes the experimental setup including dataset, metrics used followed by results. Finally, we conclude and suggest future work and extensions.

2 Proposed Approach

2.1 Pre-Processor

All the external plist information are maintained in seperate files. Each file contains phrases asssociated to that plist. For example ”distanceunit” file contains phrases like ”kilometers”, ”km”, ”miles” and so on. These plists are generally extra information that the developer has defined to aid in slot identification for low corpus training data. We load all the plist information and store it in dictionary with key as phrase and value as list of plist(s) containing the phrase. Next, we gather all the subphrases from the given utterance and search in dictionary to find any matches. Then we filter the matches based on the following procedures

- We use ibo format to tag plist for the entire utterance.For example, consider the utterance ”show me pizza stores nearby”. The corresposnding tags will be ”o o b-businesscategory i-businesscategory o” if the business category contains ”pizza stores”.

- Filter the matches based on longest match for same plist. For example if the phrase ”Barbeque Nation” is matched by BusinessName as ”Barbeque” and ”Barbeque Nation”, then we choose ”Barbeque Nation” only as final match for BusinessName and tag it in ibo format.

- Keep the matches of all plist when subphrases within the phrase are matched. For example, consider the phrase ”punjabi thali”. ”punjabi thali” is present in ”BusinessName” and ”punjabi” in ”cuisinestyle”.We keep both the plist as candidate plist in IBO format as ”b-businessname__b-cuisinestyle i-businessname”.

- If more than one plist matches the same phrase, then we concatenate them by ”__”. For example if the phrase ”Barbeque Nation” matches ”BusinessName” and ”BusinessCategory”, then we concatenate the plist as ”BusinessName__BusinessCatory” in IBO format.

The above procedure is followed to get better candidate plist result for the utterance as this information influences slot prediction for each word in the utterance. Finally, we use this external plist sequence information in the model as explained in the next sections.

2.2 BERT Model

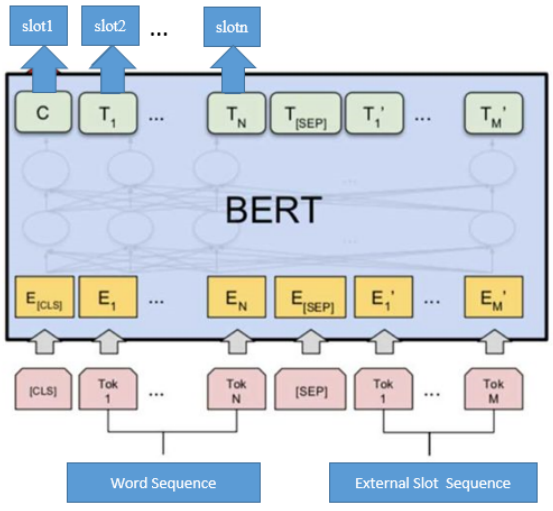


Figure 1: BERT Architecture with external plist sequence information and word sequence

Figure 1 explains BERT model architecture. BERT model is a multi-layer bi-directional Transformer encoder. The input to the model contains external plist sequence information or external slot sequence information, in addition to sentence. Since the plist can be unknown word to the vocabulary used for pre-trained BERT model, we added the list of unique plist used to the vocab list. The output of BERT provides learnt embeddings for both the word and its respective plist. To get the final embeddings for each word in sentence, we added a custom layer, which adds the embeddings of word and its plist. We pass these final embeddings to softmax layer to get the slots. We use pre-trained BERT model for the proposed experiment for the said method.

2.3 Bi-Directional Encoder with Custom Decoder

Gangadharaiah and Narayanaswamy (2019) proposed novel architecture using encoder and decoder model. Encoder module is made of Bi-directional LSTM encoder and Decoder module is made of LSTM module with attention information of input encoder during decoding stage at each step. We pass the output through dense layer. We have implemented the code from scratch based on the author description of the paper. During decoder implementation, we use attention information of encoder as additional input along with each encoder hidden information as input and calculate its importance using the modified equation as shown in Equation 6.

$$f_t = \sigma_g(W_f * x_t + U_f * h_{t-1} + V_f * a_t + b_f) \quad (1)$$

$$i_t = \sigma_g(W_i * x_t + U_i * h_{t-1} + V_i * a_t + b_i) \quad (2)$$

$$o_t = \sigma_g(W_o * x_t + U_o * h_{t-1} + V_o * a_t + b_o) \quad (3)$$

$$e_t = \sigma_g(W_c * x_t + U_c * h_{t-1} + V_c * a_t + b_c) \quad (4)$$

$$c_t = f_t * c_{t-1} + i_t * e_t \quad (5)$$

$$h_t = o_t * \sigma_c(c_t) \quad (6)$$

Where a_t represent attention information of hidden encoder h_t at time t and x_t represent input hidden encoder information at time t. We modified the input to the model to take both utterance and external plist sequence information. We converted utterance, containing list of words, to list of word indices based on dictionary (built by taking unique words from training data, sort the unique words and adding "unkword" in the end of sorted dictionary). If word is not present in dictionary, we assign "unkword" index. To maintain uniform length while training we pad sentence to max_length. We also construct weight matrix by assigning 300 dimensional vector to each row index, representing word in sorted dictionary. We obtained this 300 dimensional vector using glove embedding. If word is not in glove embedding, we assign "unkword" embedding which is randomly initialized 300 dimensional vector. We pass through Embedding module with utterance matrix and weight matrix to get 3-dimensional word embedding matrix.

We converted external plist sequence information as external slot embedding where each plist is assigned randomly initialized vector pdim (plist_dimension). As each word can have more than one plist assigned, we create matrix of plist embedding vector for each word. Hence, matrix for the external plist sequence information will be 4-dimensional (batch_size, max_sequence_words, distinct_plist_count, plist_dimension). We resize to 3D flattening the features of plist embedding from (distinct_plist_count, plist_dimension) to (distinct_plist_count * plist_dimension). We then concatenate word embedding matrix with external slot embedding matrix and pass as input to Encoder Module as shown in Figure 2.

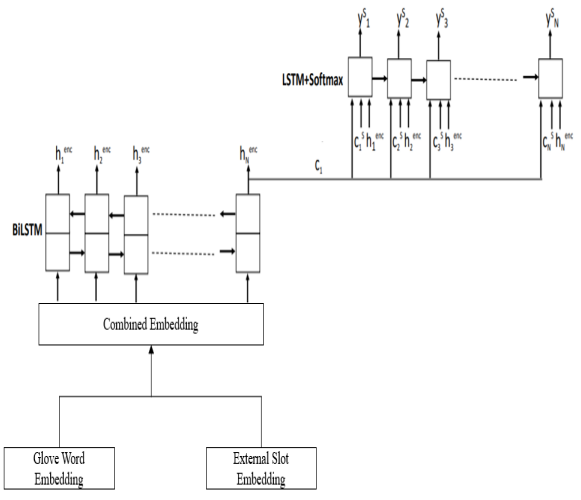


Figure 2: Bi-directional Encoder with Custom Decoder. Here we concatenate the word-embedding matrix with external slot embedding matrix and we send the combined embedding to Encoder module.

3 Experiments

Data	Train	Test	Slot
Business Search	301	1001	38

Table 1: BusinessSearch Data details.

We evaluated above models in BusinessSearch[1] Data as shown in Table 1. "BusinessSearch" data deals with details, search of Business names like dominos, reliance store and Business categories like grocery, gym with the help of external plist. It contains 301 training data and 1001 test data. It contains plist files like name, category for external use by the models.

3.1 Training Details

We share training details for BERT and Bi-directional encoder with Custom Decoder in below sections.

3.2 BERT

We used pre-trained 12 layer un-cased BERT model as initial start point for training model. We added plist names into vocab so that pre-processor does not tokenize the pist info and use it as it is. We use "Tensorflow" platform with optimizer as "Adam", loss as "categorical crossentropy", batch size as 64 and learning rate as 0.001.

3.3 Bi-directional Encoder with Custom Decoder

We use 300 dimension Glove Embedding vector for each word to construct training matrix of word index to vector. We experimented pdim with 16, 32, and 64 randomly initialized vector for each plist. We use LSTM hidden units as 128 in Encoder and Decoder (Custom LSTM with Attention information) hidden dimension as 128. Attention information does not change the hidden dimension information. Dense layer hidden dimension is distinct slots size with "Softmax" activation. We use "Keras" platform with optimizer as "Adam", loss as "categorical crossentropy", batch size as 64 and learning rate as 0.001.

4 Results and Analysis

Table 2 shows the comparison of the different architectures on "BusinessSearch" data with external plist information. From the table, we are able to infer that Bi-LSTM Encoder with Custom Decoder is able to beat BERT model where we feed external plist information as second sequence. This is attributed to the fact that concatenation of plist features along with word embedding is able to perform better than BERT model where we feed external plist information as second sequence. In addition, when we represent plist embedding dimension with 64 we are able to get better accuracy than 16 and 32. This shows that Bi-directional LSTM is able to differentiate better between different slots when plist representation is higher. In addition, we understand the importance of role of external plist as its absence lead to poor generalization of the model.

Architecture	Sentence Level Accuracy
Bi-LSTM Encoder with Decoder and 16 external slot embedding	84.79
Bi-LSTM Encoder with Decoder and 32 external slot embedding	86.83
Bi-LSTM Encoder with Decoder and 64 external slot embedding	90.49
BERT with external slot sequence as 2nd sequence	77

Table 2: Comparison of state of the art models.

5 Conclusion

This work demonstrated the use of external slot information along with sentence. We showed the performance of DNN models on low corpus data with external plist and showed there is an improvement of 13.49%, by Bi-directional Encoder with custom Decoder when compared to state of the art BERT model. We believe the pipeline approach to such user-developed dataset will aid in better model generalization for semantic slot prediction. Future scope of the paper includes exploration of non-sequential models for sequence labelling task. Also we are planning to extend the work to predict domain, intent along with slot prediction.

[1] https://github.com/MultiIntentData/LowCorpusSlotData

References

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*.

Mauajama Firdaus, Ankit Kumar, Asif Ekbal, and Pushpak Bhattacharyya. 2019. A multi-task hierarchical approach for intent detection and slot filling. *Knowledge-Based Systems*, 183:104846.

Rashmi Gangadharaiah and Balakrishnan Narayanaswamy. 2019. Joint multiple intent detection and slot labeling for goal-oriented dialog. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 564–569.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Aaron Jaech, Larry Heck, and Mari Ostendorf. 2016. Domain adaptation of recurrent neural networks for natural language understanding. *arXiv preprint arXiv:1604.00117*.

Gakuto Kurata, Bing Xiang, Bowen Zhou, and Mo Yu. 2016. Leveraging sentence-level information with encoder lstm for semantic slot filling. *arXiv preprint arXiv:1601.01530*.

Zihan Liu, Genta Indra Winata, Peng Xu, and Pascale Fung. 2020. Coach: A coarse-to-fine approach for cross-domain slot filling. *arXiv preprint arXiv:2004.11727*.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2014. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Tulika Saha, Sriparna Saha, and Pushpak Bhattacharyya. 2018. Exploring deep learning architectures coupled with crf based prediction for slot-filling. In *International Conference on Neural Information Processing*, pages 214–225. Springer.

Yangyang Shi, Kaisheng Yao, Hu Chen, Dong Yu, Yi-Cheng Pan, and Mei-Yuh Hwang. 2016. Recurrent support vector machines for slot tagging in spoken language understanding. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 393–399.

Youhyun Shin, Kang Min Yoo, and Sang-goo Lee. 2018. Slot filling with delexicalized sentence generation. In *INTERSPEECH*, pages 2082–2086.

Congrui Wang, Zhen Huang, and Minghao Hu. 2020. Sasgbc: Improving sequence labeling performance for joint learning of slot filling and intent detection. In *Proceedings of 2020 the 6th International Conference on Computing and Data Engineering*, pages 29–33.

Chenwei Zhang, Yaliang Li, Nan Du, Wei Fan, and Philip S Yu. 2018. Joint slot filling and intent detection via capsule neural networks. *arXiv preprint arXiv:1812.09471*.

Leveraging Latent Representations of Speech for Indian Language Identification

Samarjit Karmakar *
Microsoft IDC, Hyderabad
Telangana, India
karmakar.samarjit@gmail.com

P. Radha Krishna
Department of CSE
National Institute of Technology, Warangal
Telangana, India
prkrishna@nitw.ac.in

Abstract

Identification of the language spoken from speech utterances is an interesting task because of the diversity associated with different languages and human voices. Indian languages have diverse origins and identifying them from speech utterances would help several language recognition, translation and relationship mining tasks. The current approaches for tackling the problem of languages identification in the Indian context heavily use feature engineering and classical speech processing techniques. This is a bottleneck for language identification systems, as we require to exploit necessary features in speech, required for machine identification, which are learnt by a probabilistic framework, rather than handcrafted feature engineering. In this paper, we tackle the problem of language identification using latent representations learnt from speech using Variational Autoencoders (VAEs) and leverage the representations learnt to train sequence models. Our framework attains an accuracy of 89% in the identification of 8 well known Indian languages (namely Tamil, Telugu, Punjabi, Marathi, Gujarati, Hindi, Kannada and Bengali) from the CMU/IIITH Indic Speech Database. The presented approach can be applied to several scenarios for speech processing by employing representation learning and leveraging them for sequence models.

1 Introduction

Language identification refers to the task of identifying the language being spoken when given a speech utterance. Several intelligent agents rely heavily on language identification systems for subsequent speech recognition and processing tasks. This problem is particularly interesting and important in the Indian context, given the diverse nature of Indian languages. Several Indian languages suffer regional bias and each language has its own dialect as one travels within each state (region) of the country. In this scenario, language identification would be a challenging task for traditional language identification systems which heavily rely on feature engineering from speech.

Several of the current Indian language identification systems rely on handcrafted features, which end up serving as a bottleneck to such systems. Such systems would greatly benefit by learning necessary features in speech utterances using a probabilistic framework and leveraging these representations for training deep sequence models. A few deep neural network based models have also been proposed. The authors in (Lei et al., 2014) perform posterior extraction using convolutional neural networks (CNNs), and an i-vector based system for subsequent language recognition. The authors in (MounikaK. et al., 2016) use an end-to-end deep neural network with attention mechanism for Indian language identification.

We exploit the representation learnt by a VAE (Kingma and Welling, 2013) trained on speech segments to train several sequence models widely used for natural language processing. These models use distributed word-representations (Mikolov et al., 2013) which model the words in a continuous vector space. We use the latent feature representations learnt by a VAE in the stochastic low dimensional latent representation space in a manner similar to how distributed word representations, obtained by pre-training large corpuses in an unsupervised manner, are used in natural language processing to train sequence models.

In this paper, we present a framework for Indian language identification by pre-training a probabilistic framework for representation learning and leveraging these representation for training sequence models for classification.

This work was a part of the authors' undergraduate thesis project at Dept of CSE, NIT Warangal.

Proceedings of the 17th International Conference on Natural Language Processing, pages 334–340
Patna, India, December 18 - 21, 2020.

2 Related Work

Language identification from speech has been deeply studied by various research communities. Prosodic, phoenetic and phonotactic feature based approaches for identifying language is studied in (Tong et al., 2006) and (Liang Wang et al., 2006). Many such classical feature engineering based methods require a lot of domain knowledge. With the rise of deep learning and neural networks, automatic feature and representation learning has greatly outperformed all such methods.

Language identification using Deep Convolutional Recurrent Neural Networks is studied in (Bartz et al., 2017). They use non-overlapping segments of Mel spectrograms of speech which are passed through a Convolutional Neural Network and then the features maps are passed through a Long Short Term Memory (LSTM). The final hidden state of the LSTM is used for classification. The CNN captures the spatial features, whereas the LSTM captures the temporal features.

For long speech utterances, Recurrent Neural Networks, can capture the temporal aspect of speech utterances and this was considered in (Gonzalez-Dominguez et al., 2014). The authors in (Sarthak et al., 2019) give an attention based 1D-CNN for the task of language identification directly from raw audio. This attention greatly enhances the performance of neural network based approaches.

Indian language identification using deep learning based models have been studied in (Leena et al., 2005), (MounikaK. et al., 2016), (Thirumuru et al., 2018) and (Bakshi and Kopparapu, 2017). Deep neural network based systems take in the speech utterances at each frame, classification performed frame-wise, and this may be considered as a drawback. A deep neural network with attention mechanism was considered in (MounikaK. et al., 2016). This architecture applies attention to specific parts of the input sequence, whilst memorizing important features in long temporal sequences. A 39-dimensional MFCC is considered by the authors, each for 5 second chunks of the input sequence, which are passed through a regular DNN to compute hidden layer representations. An attention mechanism is applied over this to memorize the temporal aspect and summarize the features in the whole speech utterance, giving a single context vector and this vector is subsequently passed to a classifier. Attention based Residual-Time Delay Neural Network (RES-TDNN) is studied in (Mandava and Vuppala, 2019), which further improves over trying to capture the long range temporal dependencies.

3 Proposed Framework

Our goal is to learn latent representations from speech and use these representations to train sequence models for classification. We use Variational Autoencoders (VAEs) for representation learning on small segments of Mel spectrograms of speech utterances (40 Mel-scale filter banks). The Mel spectrogram is obtained by taking the Fourier transform of the signal, followed by mapping the powers of the obtained spectrum onto the Mel scale. The Mel-frequency scale resembles the resolution of the human auditory system. The segmentation is performed along the time axis. The model is trained in a similar manner adopted in (Hsu et al., 2017). After pre-training the VAE, the encoder's latent distribution is able to encode Mel spectrograms of speech segments into a latent representations space. We use a sequence of such latent representation for each segmented Mel spectrogram of speech utterance as input to sequence models. The VAE captures important representational features for each segment of the speech utterance and the sequence model captures the temporal aspect of each speech utterance. The unsupervised representation learning parameters are optimized in a different step from when the supervised sequence learning parameters are optimized.

3.1 Variational Autoencoder

A Variational Autoencoder (VAE) comprises of two neural networks, the encoder $q_\theta(z|x)$ and the decoder $p_\phi(x|z)$. The encoder, parameterized by θ, takes in the input observation (x) and encodes it into a representation (z) sampled stochastically from the distribution of μ and σ (Gaussian parametric layers of the encoder). The decoder, parameterized by ϕ, takes in the representation (z) and decodes it back into the input observation (x). The loss function (L_{vae}) minimizes a joint objective of two losses: reconstruction loss and KL divergence loss.

$$L_{vae} = -\mathbb{E}_{q_\theta(z|x)}(p_\phi(x|z)) + \mathbb{KL}(q_\theta(z|x)||p(z)) \quad (1)$$

Here, $p(z)$ is the prior distribution (multivariate standard Normal).

We use similar hyper-parameters as used in (Hsu et al., 2017). The encoder contains 3 convolutional

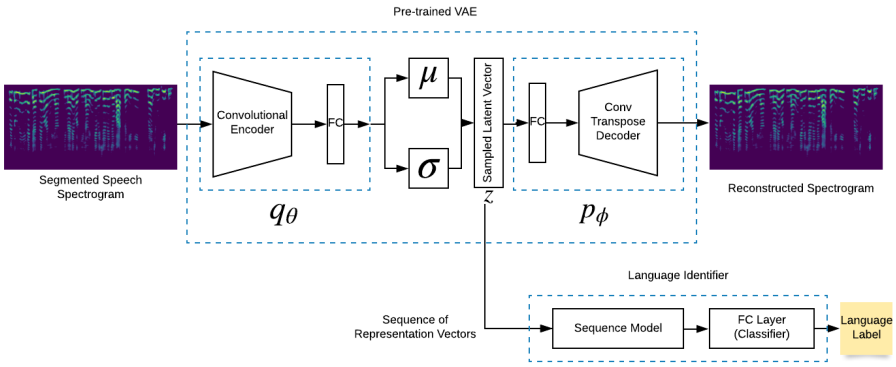


Figure 1: A view of the VAE architecture with the language identifier.

layers, followed by a fully connected layer and two Gaussian parametric layer (one for mean and another for log variance). The decoder contains an initial fully connected layer which takes in z, followed by another fully connected layer and 3 transpose-convolutional layers.

Figure 1 shows a view of the VAE architecture along with the language identifier (successor to the VAE pre-training phase).

3.2 Sequence Models

Sequence models capture the temporal aspect of the speech utterance from the given sequence of representation vectors.

For each segment of the Mel spectrogram of speech utterance, the VAE encoder produces a vector in $\mathbb{R}^n$, where n is the dimension of the representation space. We pass the sequence of these vectors for each segmented speech utterance to sequence models for classification. The input to the sequence models are a sequence representation vectors of size 128 units, i.e the dimension of the representation space of the VAE. We compute the maximum length of the sequences produced on segmentation of each speech utterance, and apply zero-padding vectors to each sequence to produce uniform length sequences.

The sequence models considered in this work are illustrated in the next sections. All the models are trained separately on the same representation vectors obtained from the pre-trained VAE encoder.

3.2.1 Long Short Term Memory (LSTM) Networks and Bi-directional LSTMs

Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) networks are sequence models which learn temporal characteristics and contextual information from sequences.

LSTMs have a single shortcoming, they make use of solely the previous context. Bidirectional LSTMs (Bi-LSTMs) make use of both the previous context as well as future context by iterating through the sequence in both directions to compute the hidden state vectors. We pass the forward and backward sequence through the LSTM assigning different weights and biases for each direction. This is used to compute two separate sets of activations for the sequence in forward and backward directions.

3.2.2 Bi-directional LSTM with Self-Attention

The sequence of representation vectors is passed through a bi-directional LSTM (bi-LSTM) (Schuster and Paliwal, 1997) with a hidden state of 100 memory units. A self-attention mechanism is adopted which gives attention to specific parts of the input sequence, giving a attention weight matrix of the input sequence. This mechanism is similar to the attention layer in (Cheng et al., 2016). The mechanism takes in the hidden states of the bi-LSTM at each time step. The attention weights are calculated for 30 sequence vectors, each giving attention to some specific part of the input sequence. The attention weights are applied to the output of the bi-LSTM, resulting in a matrix of hidden states giving attention to specific parts of the sequence. This is then flattened and passed through subsequent fully connected neural network layers to produce the output class logits.

3.2.3 Bi-directional LSTM with Soft-Aligned Attention

We apply a similar attention mechanism as that adopted in the encoder of (Bahdanau et al., 2014). We pass the sequence of representation vectors through a bi-LSTM with a hidden state of 100

memory units similar to the previous model. The difference lies in the attention mechanism adopted. We calculate a soft-alignment score between each of the hidden states of the bi-LSTM at each time step and the last hidden state. This gives the soft-aligned attention weights, which are applied to the output of the bi-LSTM at each time step to produce a single final hidden state vector. This is then passed through subsequent fully connected neural network layers to produce the output class logits.

3.2.4 Recurrent Convolutional Neural Networks

We apply a similar architecture as that adopted in (Lai et al., 2015). We pass the sequence of representation vectors through a bi-LSTM with a hidden state of 100 memory units similar to the previous model. The hidden state at each time step is concatenated to the corresponding input representation. This is passed through a fully connected layer which maps the concatenated vector back to the hidden state size. The architecture takes care of the right context and left context as it is a bi-LSTM, which takes care of information and representation flow in the forward and reverse direction of the speech utterance. We perform max-pooling across all the sequences and pass the output through subsequent fully connected neural network layers to produce the output class logits.

3.2.5 Transformer

We use the encoder of the Transformer architecture similar to that adopted in (Vaswani et al., 2017). We do not apply the positional encodings and masking mechanisms. The sequence of representation vectors are directly passed through two encoder layers, each of which which comprise of self attention and position-wise feed-forward layers. The hidden dimensions of the position-wise feed-forward layers are 100 units. The set of hyper-parameters adopted similar to (Vaswani et al., 2017) are ($N = 2$, $d_{model} = 128$, $d_q = d_k = d_v = 32$, $p_{dropout} = 0.3$). The output of the encoder is flattened and passed through subsequent fully connected neural network layers to produce the output class logits.

An illustration for training and language identification (testing) of the VAE and Sequence Model is given in Algorithm 1 and Algorithm 2.

Algorithm 1 VAE-Seq Language Identification Training

Input: Dataset D, VAE parameters (ϕ, θ), Sequence Model parameters (ξ)
Output: Optimized parameters ϕ, θ and ξ
1: Initialize parameters ϕ, θ and ξ
2: **repeat**
3: Sample mini-batch $M = \{x_i\}_{i=1,2,...\|M\|}$ of audio spectrograms from D by segmenting spectrogram of audio clips
4: Forward pass mini-batch M through VAE
5: Update parameters ϕ and θ using $\nabla_{\phi,\theta} L_{vae}(\phi, \theta, M)$
6: **until** convergence of ϕ and θ
7: **repeat**
8: Sample mini-batch $M = \{x_i, y_i\}_{i=1,2,...\|M\|}$ from D where x_i is a sequence of segmented spectrograms and y_i is label for i'th sample
9: Forward pass each sample in x_i through VAE encoder parameterized by θ to convert $\{x_i, y_i\}_{i=1,2,...\|M\|}$ to $\{v_i, y_i\}_{i=1,2,...\|M\|}$ where each v_i is a sequence of representation vectors for x_i
10: Forward pass each v_i through Sequence Model parameterized by ξ to give predicted labels $\{\hat{y_i}\}_{i=1,2,...\|M\|}$
11: Update parameters ξ using $\nabla_\xi L_{seq}(\xi, \{\hat{y_i}\}_{i=1,2,...\|M\|}, \{y_i\}_{i=1,2,...\|M\|})$
12: **until** convergence of ξ
13: **return** ϕ, θ and ξ

Algorithm 2 VAE-Seq Language Identification

Input: Speech clip x, Optimized VAE parameters (ϕ, θ), Optimized Sequence Model parameters (ξ)
Output: Language label y
1: Forward pass the segmented spectrogram of x through the VAE encoder having optimized parameters θ to obtain a sequence of representation vectors v
2: Forward pass v through the Sequence Model having optimized parameters ξ to obtain the language label y
3: **return** y

4 Experimental Results

We pre-train the VAE on segmented speech utterances from the CMU/IIITH Indic Speech Database (cmu) (Prahallad et al., 2012). The database contains raw speech utterances in 8 languages, namely Bengali, Gujarati, Hindi, Kannada, Marathi, Punjabi, Tamil and Telugu. The raw audio is converted to a Mel spectrogram (with 40 Mel filter banks and FFT window of size 1024 units). The Mel spectrogram is then segmented along the time axis, with an overlapping window of 4 units, producing a sequence of spectrograms, each of dimensions (40×20). The VAE is then trained to learn representations for these small segments (utterances) in an unsupervised manner.

Similar pre-processing is applied on each speech utterance, prior to training the sequence models, to create a sequence of spectrograms, each of dimensions (40×20), which are then passed through the pre-trained VAE encoder to produce a sequence of representation vectors, each of size 128 units.

The sequence models are trained on the above pre-processed speech data. We use cross-entropy between the output logits and the labels as the loss metric, which is minimized using Adam optimizer (Kingma and Ba, 2014), with learning rate of 10^{-4}, β_1 of 0.999, β_2 of 0.99 and weight decay

Table 1: Comparison of models

Model	Accuracy
GMM-HMM (3 languages) (Shikhamoni Nath)	86.1%
GMM + spec-pros feat (8 languages) (Vempada et al., 2013)	58.45%
DNN (8 languages) (Vuddagiri et al., 2018)	83.17%
DNN-WA (8 languages) (Vuddagiri et al., 2018)	86.10%
VAE + Bi-LSTM (1 layer) with Self-Attention	**88.24%**
VAE + Bi-LSTM (2 layers) with Self-Attention	**89.25%**
VAE + Bi-LSTM with Soft-Aligned Attention	**87.56%**
VAE + RCNN	**86.72%**
VAE + Transformer	**86.22%**

of 10^{-5}.

The results obtained on the testing set are shown in Table 1 compared with GMM-HMM based approach (Shikhamoni Nath), GMM along with spectral and prosodic features (Vempada et al., 2013), Deep Neural Network based approach (Vuddagiri et al., 2018) and DNN with Attention (Vuddagiri et al., 2018). In the table, the results are mapped to the 8 languages under consideration. The confusion matrices obtained for each sequence model are shown in Figure 2 (Appendix). We clearly see that deep learning based approaches outperform feature engineering and classical approaches. Our approach shows a performance gain in terms of accuracy compared to previous deep learning approaches as well.

5 Conclusion

In this paper, we have introduced a new framework for Indian language identification using VAE representation learning and state-of-the-art sequence models to capture the temporal characteristics of speech. The framework performs well on identification of 8 well known languages. The framework also helps improve language identification in the future as sequence models in natural language processing become better capturing long range dependencies and other temporal aspects of sequences. It can be applied in several other speech processing scenarios as well where the task requires representation learning from speech utterances and subsequent classification using a sequence model. We see VAEs are powerful probabilistic models which can learn useful representation from speech utterances and these representations can be utilized in several downstream tasks.

References

CMU Indic speech synthesis databases. http://festvox.org/cmu_indic/.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Aarti Bakshi and Sunil Kumar Kopparapu. 2017. Spoken indian language classification using artificial neural network — an experimental study. pages 424–430.

Christian Bartz, Tom Herold, Haojin Yang, and Christoph Meinel. 2017. Language identification using deep convolutional recurrent neural networks. *ArXiv*, abs/1708.04811.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561, Austin, Texas. Association for Computational Linguistics.

Javier Gonzalez-Dominguez, I. Lopez-Moreno, H. Sak, J. Gonzalez-Rodriguez, and Pedro Moreno. 2014. Automatic language identification using long short-term memory recurrent neural networks. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 2155–2159.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Wei-Ning Hsu, Yu Zhang, and James Glass. 2017. Learning latent representations for speech generation and transformation. In *Interspeech*, pages 1273–1277.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational bayes. *CoRR*, abs/1312.6114.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2267–2273. AAAI Press.

Metso Leena, K. Srinivasa Rao, and Bayya Yegnanarayana. 2005. Neural network classifiers for language identification using phonotactic and prosodic features. *Proceedings of 2005 International Conference on Intelligent Sensing and Information Processing, 2005.*, pages 404–408.

Yun Lei, Luciana Ferrer, Aaron Lawson, Mitchell McLaren, and Nicolas Scheffer. 2014. Application of convolutional neural networks to language identification in noisy conditions. In *Odyssey*.

Liang Wang, E. Ambikairajah, and E. H. C. Choi. 2006. Multi-lingual phoneme recognition and language identification using phonotactic information. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 4, pages 245–248.

L. V. D. Maaten and Geoffrey E. Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605.

Tirusha Mandava and Anil Kumar Vuppala. 2019. Attention based residual-time delay neural network for indian language identification. *2019 Twelfth International Conference on Contemporary Computing (IC3)*, pages 1–5.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

V. MounikaK., Sivanand Achanta, R. LakshmiH., Suryakanth V. Gangashetty, and Anil Kumar Vuppala. 2016. An investigation of deep neural network architectures for language recognition in indian languages. In *INTERSPEECH*.

K. Prahallad, Naresh Kumar Elluru, Venkatesh Keri, S. Rajendran, and A. Black. 2012. The iiit-h indic speech databases. In *INTERSPEECH*.

Sarthak, Shikhar Shukla, and Govind Mittal. 2019. Spoken language identification using convnets. In *European Conference on Ambient Intelligence*.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681.

Priyankoo Sarmah Samudravijaya K Shikhamoni Nath, Joyshree Chakraborty. Machine identification of spoken indian languages.

Ramakrishna Thirumuru, Ravikumar Vuddagiri, Krishna Gurugubelli, and Anil Kumar Vuppala. 2018. Significance of accuracy in vowel region detection for robust language identification. *2018 5th International Conference on Signal Processing and Integrated Networks (SPIN)*, pages 826–830.

Rong Tong, Bin Ma, Donglai Zhu, Haizhou Li, and Eng Chng. 2006. Integrating acoustic, prosodic and phonotactic features for spoken language identification. volume 1, pages I – I.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Ramu Vempada, Sudhamay Maity, and K. Rao. 2013. Identification of indian languages using multi-level spectral and prosodic features. *International Journal of Speech Technology*, 16.

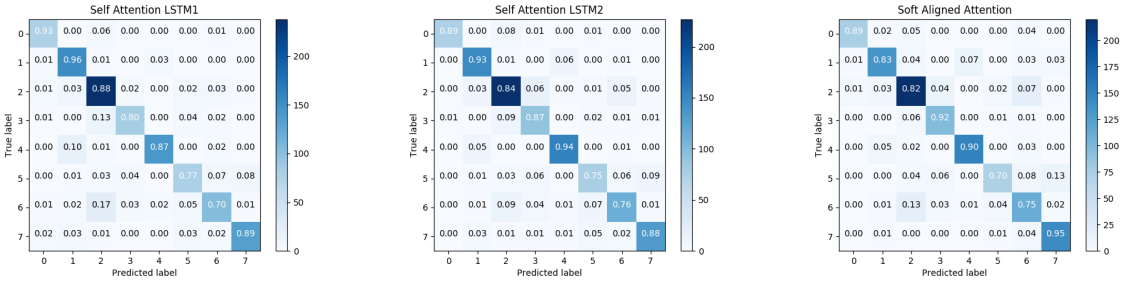
Ravi Kumar Vuddagiri, Krishna Gurugubelli, Priyam Jain, Hari Krishna Vydana, and Anil Kumar Vuppala. 2018. IIITH-ILSC Speech Database for Indain Language Identification . In *Proc. The 6th Intl. Workshop on Spoken Language Technologies for Under-Resourced Languages*, pages 56–60.

A Appendices

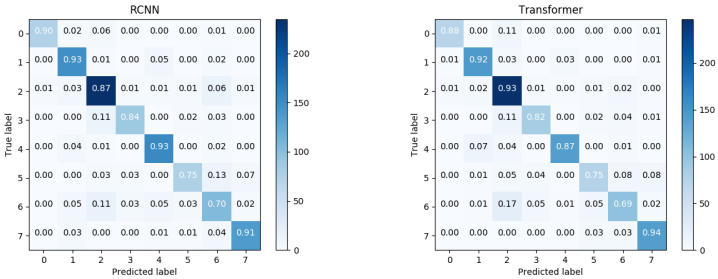
A.1 Qualitative Analysis

The T-SNE (Maaten and Hinton, 2008) embedding space of the representation in the last hidden layer assigned by each sequence model are shown in Figure 3. The visualizations give important deductions regarding the origins of each language considered.

In each T-SNE embedding space plot, we observe that Bengali and Hindi clusters appear close to each other, as Bengali and Hindi are indeed similar languages. Similar is the case with Marathi and Gujarati clusters, geographically being neighbouring states in India. The clusters of the South Indian languages of Tamil, Telugu and Kannada, geographically being neighbouring states, must appear near each other which prevails in most of the embedding space plots. The Punjabi cluster appears near the clusters of the South Indian languages of Tamil, Telugu and Kannada, an error which prevails in all the embedding space plots. There is clear distinction between the South Indian languages (believed to have Dravidian roots) and the North Indian languages (believed to have Indo-Aryan roots) in each embedding space, an important experimental finding.

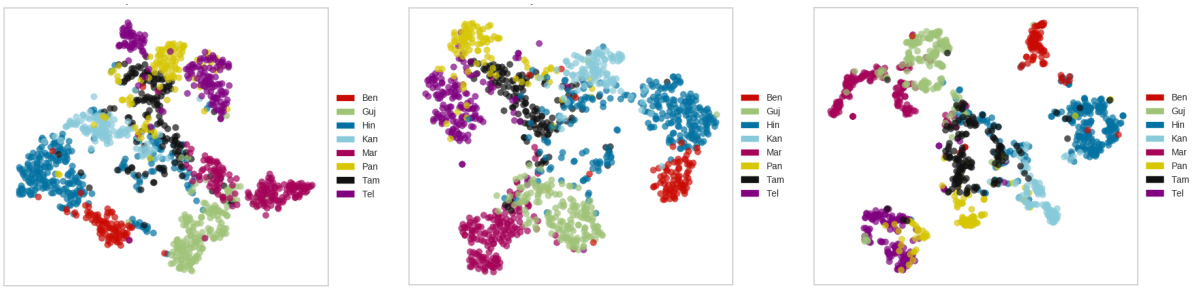


(a) Bi-LSTM with Self Attention (1 layer) (b) Bi-LSTM with Self Attention (2 layers) (c) Bi-LSTM with Soft Aligned Attention

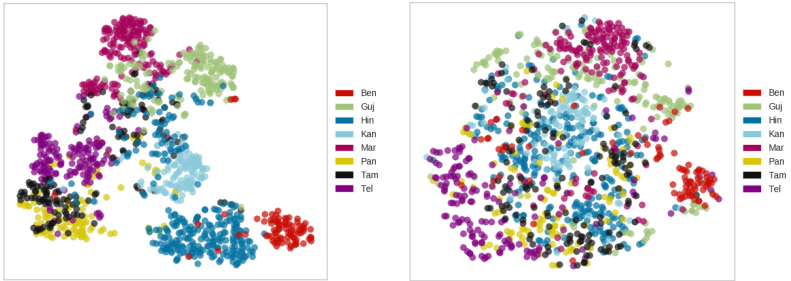


(d) RCNN (e) Transformer

Figure 2: Confusion matrices for each sequence model on test data. The corresponding labels are 0 for Bengali, 1 for Gujarati, 2 for Hindi, 3 for Kannada, 4 for Marathi, 5 for Punjabi, 6 for Tamil and 7 for Telugu.



(a) Bi-LSTM with Self Attention (1 layer) (b) Bi-LSTM with Self Attention (2 layers) (c) Bi-LSTM with Soft Aligned Attention



(d) RCNN (e) Transformer

Figure 3: T-SNE embedding space of representations in the last hidden layer assigned by each sequence model on test data.

`

Acoustic Analysis of Native (L1) Bengali Speakers' Phonological Realization of English Lexical Stress Contrast

Shambhu Nath Saha
Department of Information Technology
Narula Institute of Technology
Kolkata, India
shambhunath.saha@nit.ac.in

Shyamal Kr. Das Mandal
Centre for Educational Technology
Indian Institute of Technology
Kharagpur, India
sdasmandal@cet.iitkgp.ernet.in

Abstract

Acoustically, English lexical stress is multidimensional and involving manipulation of duration, intensity, fundamental frequency (F_0) and vowel quality. The current study investigates the acquisition of English lexical stress by L1 Bengali speakers at the phonological level in terms of the properties of acoustic cues. For this purpose, this study compares 20 L1 Bengali speakers' use of acoustic correlates for the production of English lexical stress in context sentence and neutral frame sentence. The result of this study showed that L1 Bengali speakers were not able to achieve neutral frame sentence like control over duration, intensity, F_0 and to a limited extent vowel quality in context sentence. As a result, unlike neutral frame sentence, L1 Bengali speakers were not sensitive to English lexical stress contrast in context sentence. This analysis reveals that, the difference between the neutral frame and context sentences in terms of L1 Bengali speakers' realization of phonology of English lexical stress contrast was probably due to the influence of Bengali phonology of lexical stress placement (restricted to the initial syllable of a word) on L1 Bengali speakers' English speech.

1 Introduction

Stress is one of the most important suprasegmental features in speech prosody. In linguistics, stress is the relative emphasis that may be given to certain syllables in a word, or to certain words in a phrase or sentence. English is a stress-accent language (Beckman, 1986) and English lexical stress is contrastive in nature and related to part-of-speech (Campbell and Beckman, 1997). At phonetic level, English lexical stress is acoustically related to combination of fundamental frequency (F_0), duration, intensity and vowel quality (Lieberman, 1960; Sluijter and Heuven, 1996). At phonological level, the location of English stressed syllable depends on factors such as syllable structure and lexical class. If a syllable has a long vowel, it is likely to receive primary stress, and in case of English disyllabic words, the location of stress on first or second syllable led the word to be identified as a noun or a verb respectively (Archibald, 2014; Major, 2001). As English continues to grow in importance as a language for international communication throughout the world, it is necessary for L1 Bengali speakers to acquire The English language properly. From the theory of second language acquisition, it is suggested that proper acquisition involves in correct production of one of the most important suprasegmental features that is lexical stress (Weinreich, 1979; Wode, 1978). Unlike English, word stress placement in Bengali is restricted to the initial syllable of a word (Hayes and Lahiri, 1991) and is not contrastive in nature (Chatterji, 1921). Bengali lexical stress is expressed by a combination of pitch, duration, and intensity; but stress does not affect vowel quality in Bengali (Chatterji, 1921; Emeneau, 1956). Although F_0, intensity, duration serve as cues to lexical stress in Bengali, the stress in a word is dominantly realized by a low rising pitch pattern, where the F_0 movement consists of a low F_0 valley followed by a rise (Hayes and Lahiri, 1991), and there is very little use of intensity to identify stress in Bengali (Khan, 2008).

There are fundamental differences in stress properties between English and Bengali languages at phonetic and phonological levels. At phonetic level, unlike English, vowel quality does not serve as the acoustic cue of Bengali lexical stress. Saha and Mandal (2015) previously showed that L1 Bengali speakers used the acoustic cues of vowel duration, intensity and F_0 in English like manner. Moreover, L1 Bengali speakers produced English like vowel quality in certain unstressed syllables,

Proceedings of the 17th International Conference on Natural Language Processing, pages 341–348
Patna, India, December 18 - 21, 2020.

`

but in other cases, there were significant differences in vowel quality across groups. As a result, Bengali speakers produced significantly less English like stress patterns. This was due to interference from L1 to L2 (nonnative) at the phonetic level. At phonological level, Bengali differs from English in that Bengali is bound stressed language, but the occurrence of the strongest stress at the beginning of a word is not a phenomenon appearing very commonly in English. This leads to major problems with acquiring correct stress placement habits for L1 Bengali speakers in their English speech.

The current study concentrates on the acquisition of phonology of English lexical stress placement by L1 Bengali speakers. The objective of this study is to investigate the realization of the phonology of English lexical stress by L1 Bengali speakers who were fluent in English. For this purpose, examine the differences between the uses of acoustic correlates of English lexical stress by L1 Bengali speakers under the conditions, where the position of stress to be placed in the target words in context sentence was unknown and the position of stress to be placed in the target words in neutral frame sentence was known to L1 Bengali speakers.

2 Method

2.1 Speakers

In this study, 20 L1 speakers (8 male, 12 female) of Standard Colloquial Bengali (SCB) were participated. L1 Bengali speakers were in the age group between 20 to 35 years. They were all originally from Kolkata in West Bengal and had either completed undergraduate degree studies or were continuing their postgraduate degree studies. Moreover, the L1 Bengali speakers had studied English as a second language for a minimum of ten years and were fluent in English.

2.2 Materials and Procedure

Seven pairs of disyllabic words given in Table 1 were selected following the methodology of Beckman (1986) and Fry (1955). Each word pair consisted of a noun and a verb that had identical spelling forms and differed only regarding stress placement. The words were randomly presented and were pronounced three times each by L1 Bengali speakers at their normal speech rate in the neutral frame sentence 'I said test word this time'. The stressed syllable of each target word in neutral frame sentence was marked on the reading text for speakers.

Noun	IPA Notation	Verb	IPA Notation
`contract	`kɑ:ntrækt	con`tract	kən`trækt
`desert	`dezɚt	de`sert	dI`zɚt
`object	`abʤekt	ob`ject	əb`ʤekt
`permit	`pɜ˞mIt	per`mit	pɚ`mIt
`rebel	`rebəl	re`bel	rI`bel
`record	`rekərd	re`cord	rI`kɔ:rd
`subject	`sʌbʤekt	sub`ject	səb`ʤekt

Table 1: Disyllabic words with contrasting stress positions.

Furthermore, each disyllabic target word was placed in context sentences which were shown in Table 2.

Target Word	Noun/ Verb	Context Sentence
contract	noun	Mr. Smith has finally agreed to sign the new contract.
contract	verb	Will steel contract when it is cooled?
desert	noun	They got lost in the desert.
desert	verb	Will he desert his team?
object	noun	What is the object on the table?
object	verb	They won't object to your decision.
permit	noun	In order to park here, you need a permit.
permit	verb	Would you permit her request?
rebel	noun	The rebel army did this.
rebel	verb	They rebelled at this unwelcome suggestion.
record	noun	Can I get a copy of my health record?
record	verb	She recorded all songs her daughter sang yesterday.
subject	noun	What is the subject of this sentence?
subject	verb	Must you subject me to this boring twaddle?

Table 2: Disyllabic words in context sentences with contrasting stress positions.

But stressed syllable of each target word in context sentence was not marked; i.e., speakers were not informed about the proper location of stress in the target words. The target words were randomly presented and were pronounced three times each by L1 Bengali speakers at their normal speech rate in the context sentences (Fry, 1958). The speech was recorded by using AESOP's (Visceglia et al., 2009) recording toolkit with

`

AESOP's specified recording platform. For the fluency of reading, the speakers were instructed to read out the text several times before recording and read the material aloud. The speech was digitized at a sampling rate 16 kHz with an accuracy of 16 bits/sample.

2.3 Measurements

Using Praat acoustic analysis software (Boersma and Weenink, 2004), stressed and unstressed vowels of each test word were examined acoustically for duration; the average, peak and lowest F_0; average and peak intensity. The intensity measure was calculated as the mean of multiple intensity values extracted and smoothed over the number of time points. F_0 measures were measured as the average value over entire vowel, where the pitch range for female speakers was set to 100-500 Hz and 75-300 Hz for male speakers. F1 and F2 of all vowels were measured at the middle point of their steady state and these computed formant frequencies were then averaged across the each entire vowel. The statistical analysis was done by SPSS, where two way mixed factorial analysis of variance (ANOVA) was performed with sentence type (neutral frame or context) as between subjects variable and stress position (1st syllable or 2nd syllable) as the within subjects variable for the originally measured values of each acoustic variable. All post-hoc tests (LSD) were performed with critical p value of 0.05.

3 Results and Discussions

3.1 Duration

In this study, durations of first syllable's vowel (V1) and second syllable's vowel (V2) of each test word in the neutral frame and context sentences were measured (in ms), and the results are shown in Table 3 and Table 4 and Figure 1 and Figure 2. From these results, it is observed that stressed vowels were longer than unstressed vowels in neutral frame sentence; but in the case of context sentence, L1 Bengali speakers produced stressed vowel and its unstressed counterpart with almost equal duration. Results of the analysis of vowel duration showed that there were significant main effect of sentence type [for V1: $F(1,38) = 21.34$, $p < 0.001$; for V2: $F(1,38) = 33.31$, $p < 0.001$], significant main effect of stress position [for V1: $F(1,38) = 238.82$, $p < 0.001$; for V2 : $F(1,38) = 228.27$, $p < 0.001$], as well as significant interaction between sentence type and stress position [for V1: $F(1,38) = 165.49$, $p < 0.001$; for V2 : $F(1,38) = 157.98$, $p < 0.001$]. This result indicates that there was a significant difference in the effect of stress on vowel duration between neutral frame and context sentences.

Neutral Frame Sentence		Context Sentence	
1st syllable stressed	2nd syllable stressed	1st syllable stressed	2nd syllable stressed
92.71	71.31	68.81	66.86

Table 3: Average duration of V1 (ms) in differing stress locations.

Neutral Frame Sentence		Context Sentence	
1st syllable stressed	2nd syllable stressed	1st syllable stressed	2nd syllable stressed
74.18	99.42	64.63	66.95

Table 4: Average duration of V2 (ms) in differing stress locations.

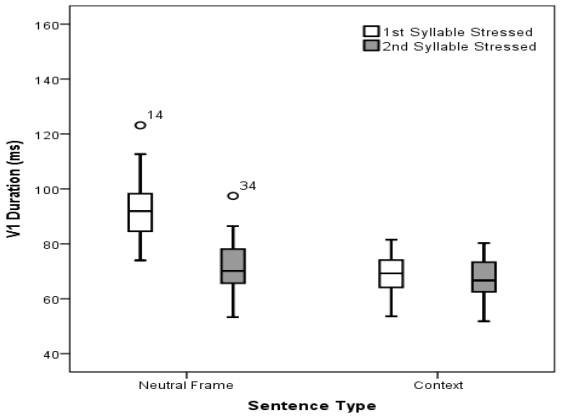


Figure 1: Average duration of V1 (ms) in differing stress locations of neutral frame and context sentences by L1 Bengali speakers.

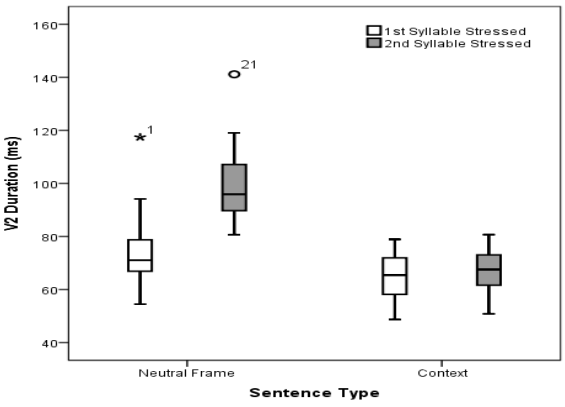


Figure 2: Average duration of V2 (ms) in differing stress locations of neutral frame and context sentences by L1 Bengali speakers.

`

The interaction effect and post-hoc test (based on sentence type) showed that there was significant difference between duration of stressed vowel and its unstressed counterpart for neutral frame sentence [V1: p = 0.000000000087, p < 0.05; V2: p = 0.00000000019, p < 0.05]; that means stressed V1 or stressed V2 was longer than unstressed V1 or unstressed V2 respectively. But in case of context sentence, there was not statistically significant difference between duration of stressed vowel and its unstressed counterpart [V1: p= 0.075, p > 0.05; V2: p = 0.08, p > 0.05]; this indicates that L1 Bengali speakers produced stressed vowel and its unstressed version of target words in context sentence with almost equal duration unlike neutral frame sentence. This was due to the influence of Bengali phonology, where the first syllable of a word is always stressed; as a result, L1 Bengali speakers' tendency was to put stress on the first syllable of each disyllabic target word in context sentence regardless of English lexical stress contrast.

3.2 Intensity

In this study, the peak and average intensity of all vowels in the disyllabic target words in the neutral frame and context sentences were measured (in dB). The ratio between V1 and V2 vowels within the same word was obtained, and the results are shown in Table 5 and Table 6 and Figure 3 and Figure 4.

Neutral Frame Sentence		Context Sentence	
1st syllable stressed	2nd syllable stressed	1st syllable stressed	2nd syllable stressed
105.05	94.62	105.03	103.25

Table 5: Average intensity ratio of V1/V2 (%) in differing stress locations.

Neutral Frame Sentence		Context Sentence	
1st syllable stressed	2nd syllable stressed	1st syllable stressed	2nd syllable stressed
104.98	94.56	105.23	103.44

Table 6: Peak intensity ratio of V1/V2 (%) in differing stress locations.

From these results, it is observed that stressed vowels were longer than unstressed vowels in neutral frame sentence; but in the case of context sentence, L1 Bengali speakers produced stressed vowel and its unstressed counterpart with almost equal duration.

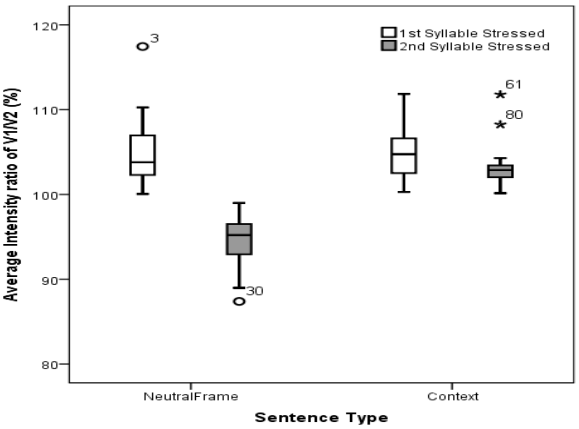


Figure 3: Average intensity ratio of V1/V2 (%) in differing stress locations of neutral frame and context sentences by L1 Bengali speakers.

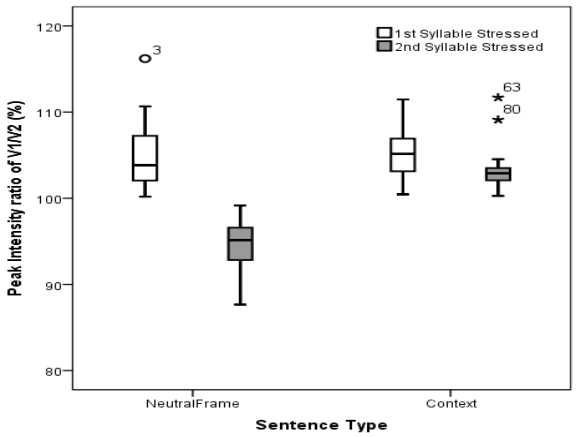


Figure 4: Peak intensity ratio of V1/V2 (%) in differing stress locations of neutral frame and context sentences by L1 Bengali speakers.

Results of the analysis of vowel duration showed that there were significant main effect of sentence type [for V1: F(1,38) = 21.34, p < 0.001; for V2: F(1,38) = 33.31,p < 0.001], significant main effect of stress position [for V1: F(1,38) = 238.82, p < 0.001; for V2 : F(1,38) = 228.27, p < 0.001], as well as significant interaction between sentence type and stress position [for V1: F(1,38) = 165.49, p < 0.001; for V2 : F(1,38) = 157.98, p < 0.001]. This result indicates that there was a significant difference in the effect of stress on vowel duration between neutral frame and context sentences. The interaction effect and post-hoc test (based on sentence type) showed that there was significant difference between duration of stressed vowel and its unstressed counterpart for neutral frame sentence [V1: p = 0.000000000087, p < 0.05; V2: p = 0.00000000019, p < 0.05]; that means

`

stressed V1 or stressed V2 was longer than unstressed V1 or unstressed V2 respectively. But in case of context sentence, there was not statistically significant difference between duration of stressed vowel and its unstressed counterpart [V1: $p = 0.075$, $p > 0.05$; V2: $p = 0.08$, $p > 0.05$]; this result indicates that L1 Bengali speakers produced stressed vowel and its unstressed version of target words in context sentence with almost equal duration unlike neutral frame sentence. This was due to the influence of Bengali phonology, where the first syllable of a word is always stressed; as a result, L1 Bengali speakers' tendency was to put stress on the first syllable of each disyllabic target word in context sentence regardless of English lexical stress contrast.

3.3 Fundamental Frequency (F_0)

The average F_0 of all vowels, peak F_0 of stressed vowels and lowest F_0 of unstressed vowels in target words were measured (in Hz) for both types of sentences. The ratio between stressed and unstressed vowels within the same disyllabic word for average F_0 and peak and lowest F_0 was obtained, and the results are shown in Table 7 and Table 8 and Figure 5 and Figure 6. From these results, it is observed that V1/V2 ratios were over 100% for average F_0 and peak/lowest F_0 when V1 or V2 was stressed for both sentence types. This result suggests that when vowels were stressed, F_0s were increased for both sentence types. Results of the analysis of average and peak/lowest F_0 ratios showed significant main effect of sentence type [for average F_0 ratio: $F(1,38) = 41.84$, $p < 0.001$; for peak F_0/lowest F_0 ratio: $F(1, 38) = 83.42$, $p < 0.001$], significant main effect of stress position [for average F_0 ratio: $F(1,38) = 43.98$, $p < 0.001$; for peak F_0/lowest F_0 ratio: $F(1,38) = 51.01$, $p < 0.001$] and significant interaction between sentence type and stress position [for average F_0 ratio: $F(1,38) = 30.93$, $p < 0.001$; for peak F_0/lowest F_0 ratio: $F(1,38) = 21.34$, $p < 0.001$]. This result implies that there was a significant difference in the effect of stress position on F_0 of vowels in disyllabic target words between the neutral frame and context sentences for L1 Bengali speakers. The interaction effect and post-hoc test (based on sentence type) showed that there was significant difference between average F_0 ratio of V1/V2 [$p = 0.00000000018$, $p < 0.05$] as well as the ratio between peak and lowest F_0s [$p = 0.00000000044$, $p < 0.05$] in differing stress locations for neutral frame sentence. This result indicates that, for neutral frame sentence, the average F_0 ratio of V1/V2 and peak and the lowest F_0 ratio of V1/V2 were significantly higher when V1 was stressed compared to V2 was stressed. But there was not statistically significant difference between average F_0 ratio of V1/V2 [$p = 0.45$, $p > 0.05$] as well as the ratio between peak and lowest F_0s [$p = 0.083$, $p > 0.05$] in differing stress locations for context sentence. This result implies that V1/V2 ratio was almost equal in differing stress locations for average F_0 ratio as well as peak and lowest F_0 ratio for context sentence, unlike neutral frame sentence.

Neutral Frame Sentence		**Context Sentence**	
1st syllable stressed	**2nd syllable stressed**	**1st syllable stressed**	**2nd syllable stressed**
117.4	109.65	105.6	104.92

Table 7: Average F_0 ratio of V1/V2 (%) in differing stress locations.

Neutral Frame Sentence		**Context Sentence**	
1st syllable stressed	**2nd syllable stressed**	**1st syllable stressed**	**2nd syllable stressed**
122.1	112.22	106.57	104.46

Table 8: Peak & lowest F_0 ratio of V1/V2 (%) in differing stress locations.

That means the increase in F_0 of V1 was significantly higher than that of V2 in the same disyllabic target word of context sentence in differing stress locations. From this result, it is revealed that, due to the interference of Bengali phonology, V1 was always stressed instead of V2 of the same disyllabic target word in context sentence by L1 Bengali speakers regardless of recognizing English lexical stress contrast.

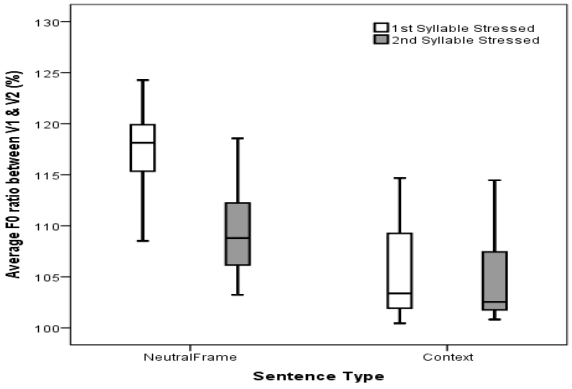


Figure 5: Average F_0 ratio of V1/V2 (%) in differing stress locations of neutral frame and context sentences by L1 Bengali speakers.

`

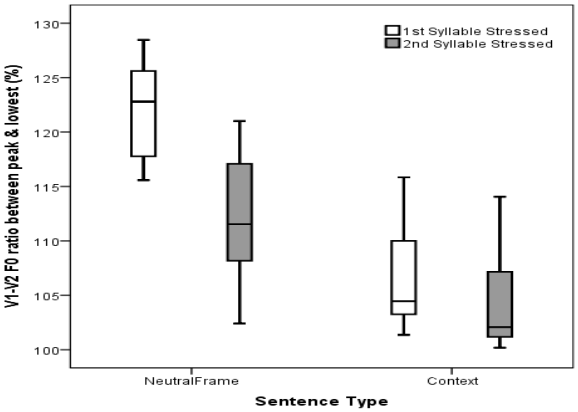


Figure 6: Peak and lowest F_0 ratio of V1/V2 (%) in differing stress locations of neutral frame and context sentences by L1 Bengali speakers.

3.4 Vowel Quality

Vowel quality is defined in terms of first (F1) and second (F2) formant frequencies (Kul, 2010). In this study, formant spacing was used to quantify the property of vowel quality, where two measures are derived from the center frequencies of F1 and F2 (Blomgren et al., 1998; Amir and Amir, 2007). The compact-diffuse (C-D), calculated as the difference between F1 and F2 (F2-F1), is correlated with the phonetic property of tongue height. The grave-acute (G-A) feature, calculated as the arithmetic mean of F1 and F2 [(F1+F2)/2], is correlated with the phonetic dimension of the tongue advancement. For each syllable in each word, separate ANOVAs were performed for both C-D and G-A variables with two factors – sentence type (between subjects variable) and stress position (within-subjects variable). Results of post-hoc test (LSD) at level $p < 0.05$ are shown in Table 9, where S refers to stressed syllables, U to unstressed syllables, NF to production of neutral frame sentence and C to production of context sentence by L1 Bengali speakers. NF < C (NF > C) indicates that Bengali speakers' productions of a given syllable in neutral frame sentence showed smaller (higher) mean values of a given acoustic feature than did context sentence.

Similarly, S < U (S > U) indicates smaller (higher) mean values of a given acoustic feature for stressed syllable compared to that of unstressed syllable for the corresponding type of sentence. From the result of analysis (shown in Table 9), it is observed that Bengali speakers did not show statistically significant difference in their production of most of the stressed as well as unstressed syllables between the neutral frame and context sentences. Only exceptions were for the syllables *-tract* (*contract*), *-ject* (*object*), *-mit* (*permit*) and *–re* (*record*), in which the stressed or unstressed or both versions did show the significant difference between the neutral frame and context sentences in terms of C-D or G-A or both features. Overall five general patterns were found from this analysis:

Type 1. Correct non-reduction: L1 Bengali speakers did not reduce the vowel in the following unstressed syllables of both neutral frame and context sentences (no significant differences were found for either C-D or G-A): *de-* (*desert*), *ob-*(*object*), *-cord* (*record*).

Type 2. Lack of reduction: Unlike neutral frame sentence, there was not found a significant change in either C-D or G-A features from stressed to unstressed versions of the following syllables of context sentence: *con-* (*contract*), *-mit* (*permit*), *re-* (*rebel*).

Type 3. Unexpected reduction: Unlike neutral frame sentence, L1 Bengali speakers significantly reduced unstressed vowel (in terms of either C-D or G-A or both) in the following syllables of context sentence: *per-* (*permit*), *re-* (*record*).

Type 4. Incorrect reduction: In these syllables of both neutral frame and context sentences, there were significant differences between stressed and unstressed vowels, but the unstressed vowel of context sentence was in each case significantly different (in terms of either C-D or G-A, or both)from its neutral frame counterpart. These syllables include: *-tract (contract), -ject (object)*.

Type 5. Correct reduction: Syllables in which both neutral frame and context sentences show the significant difference between stressed and unstressed vowels. Moreover, there was not statistically significant difference between unstressed vowels of the neutral frame and context sentences (in terms of either C-D or G-A or both). These syllables include: *-sert (desert), -bel (rebel), sub- (subject), -ject (subject)*.

Based on these comparisons, it reveals that L1 Bengali speakers showed lots of similarity between the neutral frame and context sentences regarding stressed and to a limited extent unstressed vowel productions.

`

Syllable	Stressed / Unstressed				Neutral Frame Sentence/Context Sentence			
	Neutral Frame Sentence		Context Sentence		Stressed		Unstressed	
	C-D	G-A	C-D	G-A	C-D	G-A	C-D	G-A
con-	S < U							
-tract	S < U	S > U		S > U	NF < C		NF < C	
de-								
-sert	S < U	S < U	S < U					
ob-								
-ject	S < U		S > U				NF > C	
per-				S < U				
-mit		S > U				NF > C		
re-	S < U							
-bel		S > U		S > U				
re-			S < U	S < U	NF > C	NF > C		
-cord								
sub-	S < U		S < U	S > U				
-ject		S > U	S > U					

Table 9: Results of pair wise comparisons between formant measures for stressed and unstressed vowels by syllable.

For most syllables, stressed vowels did not show a significant difference between the neutral frame and context sentences (Table 9, fifth and sixth columns); the exceptions were *-tract (contract)*, *-mit (permit)*, and *re- (record)*. This means that, for the majority of vowels used in the stressed syllable, L1 Bengali speakers employed approximately the similar quality and category of vowels in both neutral frame and context sentences. Furthermore, there was not statistically significant difference in unstressed vowels of most syllables between the neutral frame and context sentences (Table 9, seventh and eighth columns), with the exceptions of *–tract (contract)* and *–ject (object)*. This observation indicates that L1 Bengali speakers produced similar degree and quality of reduced vowels in most of the syllables of both neutral frame and context sentences.

4 Conclusions

From the results of this study, it appears that L1 Bengali speakers showed a substantial difference in use of the acoustic correlates of vowel duration, intensity, and F_0 of English lexical stress between the neutral frame and context sentences. But, L1 Bengali speakers did not show the significant difference in vowel quality of stressed syllable between the neutral frame and context sentences and L1 Bengali speakers reduced vowel in the unstressed syllable with a similar degree and quality in both types of sentences. This acoustic analysis reveals that L1 Bengali speakers were not able to achieve neutral frame sentence like control over duration, intensity, F_0 and to some extent quality of stressed and unstressed vowels in context sentence. This difference between both types of sentences was probably due to the interference from Bengali phonology of lexical stress placement on L2 English, where L1 Bengali speakers' tendency was to put stress on the first syllable of each disyllabic target word in context sentence without recognizing its lexical stress contrast. Hence, results of this acoustic analysis suggest that, although L1 Bengali speakers were able to produce lexical stress contrast in neutral frame sentence in an English-like manner, they were not sensitive to English lexical stress contrast in context sentence.

References

Mary. E. Beckman. 1986. *Stress and non-stress accent*, *volume 7*.Walter de Gruyter.

Nick Campbell and Mary. E. Beckman. 1997. Stress, prominence, and spectral tilt. In *Proceedings of ESCA Workshop on Intonation: Theory, Models and Applications*, pages 67–70.

Philip Lieberman. 1960. Some acoustic correlates of word stress in American English. *The Journal of the Acoustical Society of America*, 32(4): 451-454.

Agaath. M. Sluijter and Vincent. J. Van Heuven.1996. Spectral balance as an acoustic correlate of linguistic stress. *The Journal of the Acoustical society of America*, 100(4): 2471-2485.

`

John. Ed. Archibald. 2014. *Phonological acquisition and phonological theory*. Psychology Press.

Roy. C. Major. 2001. *Foreign accent: The ontogeny and phylogeny of second language phonology*. Routledge.

Uriel Weinreich.1979. *Languages in contact: Findings and problems*. Walter de Gruyter.

Henning Wode. 1978. The beginnings of non-school room L2 phonological acquisition. *IRAL-International Review of Applied Linguistics in Language Teaching*, 16(1-4):109-126.

Bruce Hayes and Aditi Lahiri. 1991. Bengali intonational phonology. *Natural Language & Linguistic Theory*, 9(1): 47-96.

Suniti. K. Chatterji. 1921. Bengali Phonetics. *Bulletin of the School of Oriental and African Studies*, 2:1-25.

Murray. B. Emeneau.1956. India as a linguistic area. *Language*, 32(1): 3-16.

Sameer. U.D. Khan. 2008. Intonational Phonology and Focus Prosody in Bengali (PhD Thesis). *University of California,* Los Angeles.

Shambhu. N. Saha and Shyamal. K. Das. Mandal. 2015. Study of Acoustic Correlates of English Lexical Stress Produced by Native (L1) Bengali Speakers Compared to Native (L1) English Speakers. In *Proceedings of Annual Conference of the International Speech Communication Association*, pages 815-819.

Dennis. B. Fry. 1955. Duration and intensity as physical correlates of linguistic stress. *The Journal of the Acoustical Society of America*, 27(4):765-768.

Dennis. B. Fry. 1958. Experiments in the perception of stress. *Language and speech*, l (2):126-152.

Tanya Visceglia, Chiu-yu Tseng, Mariko Kondo, Helen Meng, and Yoshinori Sagisaka.2009. Phonetic aspects of content design in AESOP (Asian English Speech cOrpus Project). In *Oriental COCOSDA International Conference on Speech Database and Assessments*, pages 60-65.

Paul Boersma and David Weenink. 2004. http://www.fon.hum.uva.nl/praat/.

Małgorzata Kul. 2010. Towards a gradual scale of vowel reduction: a pilot study. *Poznan Studies in Contemporary Linguistics,* 46(4): 429-456.

Michael Blomgren, Michael Robb, and Yang Chen. 1998. A note on vowel centralization in stuttering and nonstuttering individuals. *Journal of Speech, Language, and Hearing Research*, 41(5): 1042-1051.

Noam Amir and Ofer Amir. 2007. Novel measures for vowel reduction. In *Proceedings of the 16th International Congress of Phonetic Sciences*, pages 849-852.

Towards Performance Improvement in Indian Sign Language Recognition

Kinjal Mistree
Computer Engineering Department
Uka Tarsadia University
Bardoli
kinjal.mistree@utu.ac.in

Devendra Thakor
Computer Engineering Department
Uka Tarsadia University
Bardoli
devendra.thakor@utu.ac.in

Brijesh Bhatt
Computer Engineering Department
Dharmsinh Desai University
Nadiad
brij.ce@ddu.ac.in

Abstract

Sign language is a complete natural language used by deaf and dumb people. It has its own grammar and it differs with spoken language to a great extent. Since people without hearing and speech impairment lack the knowledge of the sign language, the deaf and dumb people find it difficult to communicate with them. The conception of system that would be able to translate the sign language into text would facilitate understanding of sign language without human interpreter. This paper describes a systematic approach that takes Indian Sign Language (ISL) video as input and converts it into text using frame sequence generator and image augmentation techniques. By incorporating these two concepts, we have increased dataset size and reduced overfitting. It is demonstrated that using simple image manipulation techniques and batch of shifted frames of videos, performance of sign language recognition can be significantly improved. Approach described in this paper achieves 99.57% accuracy on the dynamic gesture dataset of ISL.

1 Introduction

According to (Durkin and Conti-Ramsden, 2010), sign language can be considered as a collection of gestures, movements, postures, and facial expressions corresponding to letters and words in spoken languages. Comparing sign language with spoken language, main difference exists on modality. Deaf and dumb people use sign language as means of communication to express their thoughts and emotions. Each country has its own sign language with high degree of grammatical variations. Indian sign language is largely dominated by object features, and is different than other sign languages.

Trained sign language interpreters are needed during medical and legal appointments, educational and training sessions, to provide interpreting services. Over the past few years, there has been an increasing demand for human interpreters. But in India, approximately 300 certified human interpreters are available (vid). Given a shortage of interpreting services, a system can be designed that offers flexible alternative when human interpreters are not available.

Sign language recognition is an approach that converts input sign gesture(s) into text or speech. Sign language recognition is a very challenging task since this task involves interpretation between visual and linguistic information. Deep neural networks (DNN) perform remarkably well for image recognition task (Shorten and Khoshgoftaar, 2019). But these networks are heavily reliant on big data, otherwise they lead to overfitting. Overfitting refers to the phenomenon when a network learns a function with very high variance such as to perfectly model the training data (Shorten and Khoshgoftaar, 2019). Strategies to avoid overfitting and to increase generalization performance of DNNs are dropout, batch normalization, transfer learning, one-shot learning and pretraining. In contrast to these techniques, data augmentation approaches problem of overfitting and generalization from the root of the problem, the training dataset. We have adopted this concept on input videos for ISL recognition to make dataset inflated.

In order to increase dataset size, we have created sequence of frames in batches systematically. We have proposed this approach to generate more instances for ISL recognition task in order to achieve better model performance. In particular, we have addressed the issue of one research question: how to use DNN with very small amount of input videos in order to incorporate both left-handed and right-handed signs without hurting recognition performance of ISL sentences.

The rest of the paper is organized as follows: In Section 2 we have discussed work related to Indian sign language recognition. Section 3 explains steps of our proposed approach in details. Section 4

Proceedings of the 17th International Conference on Natural Language Processing, pages 349–354
Patna, India, December 18 - 21, 2020.

shows dataset description and experimental results along with analysis. Section 5 provides concluding remarks and directions for future work.

2 Related Work

There are two approaches for sign language recognition: 1) Device based approach and 2) Vision based approach. Device based approach hinders the natural movement of hands. Also, signer has to wear gloves that are connected through computer through cables. So device based approaches appear to be complex, costly and difficult to deploy. Considering limitations of device based approach, we have discussed related work with respect to vision based approach in ISL recognition.

(Rekha et al., 2011) proposed an approach that recognized 66 ISL dynamic samples of ISL alphabets. These gestures were classified using Dynamic Time Warping and approach achieved 77.2% accuracy. (Tripathi and Nandi, 2015) presented an approach for recognizing ISL sentences in ISL with 91% accuracy with Hidden Markov Model. Authors recorded single handed and double handed dynamic samples from 10 sentences. Recognition rate of 90.17% was achieved by (Kishore et al., 2016) for 580 samples. Though these samples were not recorded according to ISL grammar, this approach worked for manual components of ISL. (Kumar et al., 2016) used front camera of the mobile phone for collecting dynamic signs. The authors achieved 90% accuracy by extracting hand and head contour energies from the collected dynamic signs.

Issues like hand segmentation from upper half of the body image, boundary changes depending on hand shape of various signers are solved by (Baranwal and Nandi, 2016) with 85% accuracy. Authors have used Otsu thresholding method for segmentation, Mel Sec Frequency Cepstral Coefficients (MFCC) for feature extraction of 8 dynamic samples. 92% accuracy was achieved by (Baranwal et al., 2017) using concept of possibility theory on 20 different videos of continuous gestures. These videos were captured in different backgrounds like black, red, multiple objects with black full sleeves dress. (Wazalwar and Shrawankar, 2017) used pseudo 2-dimensional Hidden Markov Model (P2DHMM) for feature extraction, which was proven better than simple Hidden Markov Model. For converting recognized signs in English text, LALR parser was used and 90% accuracy was achieved. (Muthu Mariappan and Gomathi, 2019) proposed an approach that extracted head and hand contour energies using front camera of the mobile for collecting signs.Approach presented by (Sahoo and Ravulakollu, 2014) used skin color segmentation and artificial neural network (ANN). This approach worked for specific signer as ANN was trained by taking face and hand features of specific signer.

As far as reserach in ISL is concerned, after nearly 35 years of research, ISL is still in infancy when compared to other international sign languages. All the approaches described here work under controlled laboratory setting and use feature extraction techniques. This motivated us to work for ISL recognition by doing generalized settings in very limited amount of input data without decreasing accuracy of ISL recognition.

3 Method

The details of the dataset are described in the Section 4. In this section we describe the steps of the approach we followed to convert ISL video into text. Figure 1 shows the general framework of our approach.

3.1 Video to frame conversion

Each ISL input video is converted in RGB frames. Videos were originally captured at 30 fps, so accordingly frames are generated for each video of different length.

3.2 Horizontal flipping

Signer is either left-handed or right-handed. On the batch of frames, horizonal flipping is performed in order to incorporate both left-handed and right-handed signs.

3.3 Frame sequence generator with data augmentation

To identify main frames, one possibility is that we pick N distributed frames from the entire video. But this works only with fixed length of video as we may lose important information from frames. To address this issue, we have created a video generator that provides decomposed frames for the entire video in sequential form to get more sequences from one video file. For each 30 FPS video we have used this video generator to select 5 frames per second. We have decided to select every 6^{th} frame based on analysis of histogram difference in frames. For each individual video, frames are

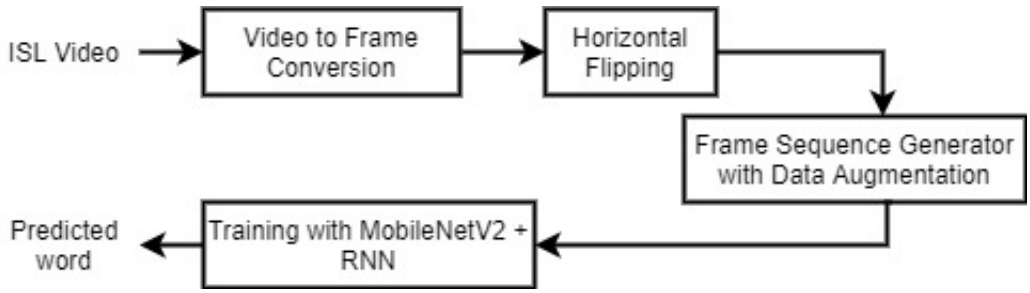


Figure 1: General framework of frame sequence generator based approach

selected in batches in order to get a set of shifted frames, such as first batch has frames 1, 7, 13, 19, 25 in sequence; second batch has frames 2, 8, 14, 20, 26 in sequence and so on.

This custom generator supports image augmentation techniques. On the resultant images after frame sequence generator, geometric transformations- zooming, rotation, vertical shifting, horizontal shifting; and photometric transformations, augmentation on brightness are performed.

(Perez and Wang, 2017) have discussed how to produce promising ways to increase the accuracy of classification tasks using data augmentation. We have decided to work with augmentation techniques based on two aspects: various video recording conditions and hardware dependency. For end-to-end ISL recognition, the environment in which signers perform signs under lighting and camera settings may be different. Signers may use different hardware devices such as camera, smartphone, tablets, computer with different resolutions and view. These variances are addressed by training the deep learning model with randomly selected augmentation types within range of parameters. We have shown that training the recognizer with inflated data with randomness in augmentation gives remarkable improvement in accuracy. Image augmentation types and parameters were randomly selected with frame sequence generator.

3.4 Training with MbileNetV2 + RNN

Image augmentation increases the size of the dataset which is originally very small but the data similarity is still very high. Transfer learning works well with limited and similar data samples by transferring knowledge from models pretrained on large datasets. Among the popular pretrained models, we have used MobileNetV2 as it is light-weight, low-latency deep neural network best suited with restricted resources in mobile and embedded vision applications (Sandler et al., 2018). We have empirically changed the configuration of the top layers of the MobileNetV2 model in order to get the best recognition accuracy. Based on this, top 9 layers of the model are selected for retraining with the augmented frame sequence. This is injected in one time-distributed layer at the end to have the one-dimensional shape compatible with LSTM layer. Finally, dense layer is added to get the prediction of ISL word.

4 Experimental Results and Analysis

In this section, we present the details of the dataset and the experimental results with analysis.

4.1 Dataset

(Nandy et al., 2010) created repository of static and dynamic hand gestures of 21 specific kind of ISL words under various light illumination conditions. Out of 21 classes, 11 classes corresponds to dynamic hand gestures and 10 classes corresponds to static hand gestures. The dataset was created in July 2009 at the Robotics and AI Lab, IIIT-Allahabad, having frame resolution of 320 * 240 pixels. Statistics of training, validation and testing samples used in work are shown in Table 1.

4.2 Results and analysis

Training and testing data used by us was prepared under various light illumination conditions but with identical camera settings. As discussed in previous section, we have chosen parameter range in order to incorporate randomness in sample generation. We have excluded horizontal flipping from this set of augmentation techniques because we already inflated dataset in order to incorporate both left-handed as well as right-handed signs. Table 2 shows type of augmentation techniques and parameter range used for our experiments.

Parameters	Values
No. of classes (ISL words)	10
Training samples	79
Validation samples	15
Testing samples	27
Training samples after horizontal flipping	158
Validation samples after horizontal flipping	30
Testing samples after horizontal flipping	54
No. of training sequences after using frame sequence generator	12692
No. of validation sequences after using frame sequence generator	2443
No. of testing sequences after using frame sequence generator	4082

Table 1: Image sequences after using frame sequence generator with data augmentation

Figure 2: Sample frames 5, 11, 17, 23 and 29 for sign 'below' as per selection by frame sequence generator. This frame set is result of frame sequence generator + image augmentation techniques.

Figure 3: Another set of frames 2, 8, 14, 20 and 26 for sign 'below' as result of frame sequence generator + image augmentation techniques.

Figure 4: Sample frames 3, 9, 15, 21 and 27 for sign 'below' as per selection by frame sequence generator. This frame set is result of horizonal flipping + frame sequence generator + image augmentation techniques.

Augmentation Type	Parameter Range
Zooming	[0.7, 1]
Rotation	[0, 15]
Vertical shifting	[0, 0.3]
Horizontal shifting	[0, 0.3]
Brightness	[0.2, 1.0]
Shear angle	[0, 0.3]

Table 2: Augmentation techniques and range of parameters used for each ISL input video

Figure 2, 3 and 4 shows various frame sequences generated by video generator for sign 'below', using frame sequence generator.

By following the steps explained in the previous section, an experiment was conducted on ISL dynamic gesture dataset for 10 categories of ISL words. Table 3 shows model performance on ISL dynamic gesture dataset using MobilNetV2 + RNN, by keeping top 6, 9 and 12 layers trainable, while keeping other layers of model frozen. We have achieved recognition accuracy as 99.57% when keeping last 9 layers trainable, which outperforms the overall accuracy reported by (Nandy et al., 2010).

Figure 5 and Figure 6 shows plot of accuracy and

MobileNetV2+RNN	Trainable layers = 6	Trainable layers = 9	Trainable layers = 12
Training accuracy	98.33%	99.41%	93.12%
Validation accuracy	98.67%	100%	93.85%
Testing accuracy	93.91%	99.57%	94.15%
Training loss	0.2134	0.0313	0.3874
Validation loss	0.0976	0.0010	0.2916
Testing loss	0.0841	0.0016	0.0287

Table 3: Model performance on ISL dynamic gesture dataset

loss, by keeping last 9 and layers as trainable. We have trained model for 40 epochs and used early stopping on validation loss, so the training gets stopped when there is no significant improvement in accuracy after 3 continuous epochs.

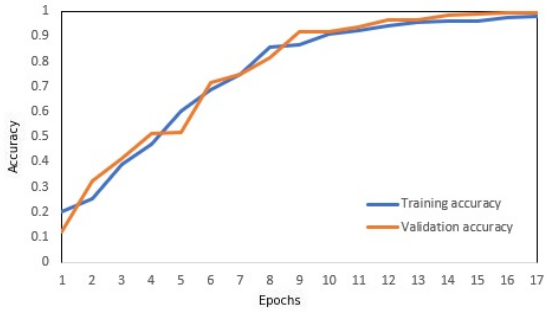


Figure 5: Training and validation accuracy by keeping last 9 layers trainable using MobileNetV2+RNN

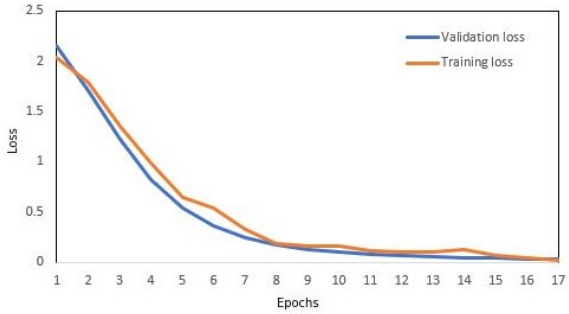


Figure 6: Training and validation loss by keeping last 9 layers trainable using MobileNetV2+RNN

Approach	Accuracy
(Nandy et al., 2010)	81.94%
Our method	99.57%

Table 4: Accuracy of proposed approach on ISL dynamic gesture dataset

Table 4 shows comparison of recognition result in terms of accuracy, for approach presented by (Nandy et al., 2010) and our approach. In previous work, results for 11 ISL words are presented. We have excluded result of class 'Yes' as dataset provided by authors has not sufficient samples for sign 'Yes'. Also, we have used less number of training, validation and testing samples in order to evidently prove the effect of our proposed approach in classification.

5 Conclusion and Future work

It becomes a challenging task when we want to achieve more accuracy with less number of samples in generalized environment. Deep learning gives promising results than other traditional algorithms in computer vision task as they learn features from gestures, but they require huge dataset. To overcome the problem of overfitting generated by deep learning models on less amount of data, image augmentation can be used before training data. Image augmentation also increases accuracy of test data. In this work, we have empirically proven that simple image manipulation techniques and pretrained model with frame sequence generator creates great impact on the accuracy on ISL recognition than using very limited amount of data in training. We have proposed an approach that uses pretrained model MobileNetV2 to learn features from augmented frame sequences of ISL gestures using batch of shifted frames to provide decayed sequences for the same gesture.

We are working on extending our work from lexical level analysis to machine translation to generate ISL sentences. We are also in the process of creating new dataset of ISL sentences using the dictionary launched by Indian Sign Language Research and Training Centre (ISLRTC). In future, we will compare results of MobileNetV2 model with other pretrained models on our dataset.

References

www.islrtc.nic.in/history-0. Accessed April 28, 2019.

Neha Baranwal and G. Nandi. 2016. An efficient gesture based humanoid learning using wavelet descriptor and mfcc techniques. *International Journal of Machine Learning and Cybernetics*, 8.

Neha Baranwal, Avinash Singh, and G. Nandi. 2017. Development of a framework for humanrobot interactions with indian sign language using possibility theory. *International Journal of Social Robotics*, 9.

Kevin Durkin and Gina Conti-Ramsden. 2010. Young people with specific language impairment: A review of social and emotional functioning in adolescence. *Child Language Teaching & Therapy - CHILD LANG TEACH THER*, 26:105–121.

P.V.V. Kishore, M.V.D. Prasad, D. Anil Kumar, and A Sastry. 2016. Optical flow hand tracking and active contour hand shape features for continuous sign language recognition with artificial neural networks.

D. A. Kumar, P. V. V. Kishore, A. S. C. S. Sastry, and P. R. G. Swamy. 2016. Selfie continuous sign language recognition using neural network. In *2016 IEEE Annual India Conference (INDICON)*, pages 1–6.

H. Muthu Mariappan and V. Gomathi. 2019. Real-time recognition of indian sign language. In *2019 International Conference on Computational Intelligence in Data Science (ICCIDS)*, pages 1–6.

Anup Nandy, Jay Prasad, Soumik Mondal, Pavan Chakraborty, and G. Nandi. 2010. Recognition of isolated indian sign language gesture in real time. volume 70, pages 102–107.

Luis Perez and Jason Wang. 2017. The effectiveness of data augmentation in image classification using deep learning.

J. Rekha, J. Bhattacharya, and S. Majumder. 2011. Shape, texture and local movement hand gesture features for indian sign language recognition. In *3rd International Conference on Trendz in Information Sciences Computing (TISC2011)*, pages 30–35.

Ashok Sahoo and Kiran Ravulakollu. 2014. Indian sign language recognition using skin colour detection. *International Journal of Applied Engineering Research*, 9:7347–7360.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks.

Connor Shorten and Taghi M. Khoshgoftaar. 2019. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6:1–48.

Kumud Tripathi and Neha Nandi. 2015. Continuous indian sign language gesture recognition and sentence formation. *Procedia Computer Science*, 54:523–531.

Sampada Wazalwar and Urmila Shrawankar. 2017. Interpretation of sign language into english using nlp techniques. *Journal of Information and Optimization Sciences*, 38:895–910.

Question and Answer Pair Generation for Telugu Short Stories

Meghana Bommadi, Shreya Terupally, Radhika Mamidi
Language Technologies Research Centre
International Institute of Information Technology Hyderabad, India
meghana.bommadi@research.iiit.ac.in , shreya.reddy@students.iiit.ac.in,
radhika.mamidi@iiit.ac.in

Abstract

Question Answer pair generation is a task that has been worked upon by multiple researchers in many languages. It has been a topic of interest due to its extensive uses in different fields like self assessment, academics, business website FAQs etc. Many experiments were conducted on Question Answering pair generation in English, concentrating on basic Wh-questions with a rule-based approach. We have built the first hybrid machine learning and rule-based solution in Telugu which is efficient for short stories or short passages in children's books. Our work covers the fundamental question forms with the question types: adjective, yes/no, adverb, verb, when, where, whose, quotative, and quantitative(how many/ how much). We constructed rules for question generation using POS tags and UD tags along with linguistic information of the surrounding context of the word.

1 Introduction

Question and Answer pair generation is an open problem in linguistics which deals with Natural Language Understanding (NLU) and Natural Language Generation (NLG). NLU and NLG are commonly used in interactive NLP applications such as AI-based dialogue systems/voice assistants like SIRI, Google Assistant, Alexa, and similar other personal assistants. Numerous methods are introduced for the Q&A pair generation problem. For a low-resourced language like Telugu, AI-based solutions can be non-viable. There are hardly any datasets available for the system to produce significant accuracy. A completely rule-based system might leave out principle parts of the abstract. There is a chance that all the questions cannot be captured inclusively by completely handwritten rules. Hence, we wanted to introduce a mixed rule-based and AI-based solution to this problem.

We attempted to produce questions, concentrating on the key points of a text that are generally asked in assessment tests. Questions posed to an individual challenge their knowledge and understanding of specific topics, so we formed questions in each sentence in as many ways as possible. We based this paper on children's stories, so the questions we wanted to produce aim to be simpler and more objective.

Based on the observation of the data chosen and after analyzing all the possible cases, we developed a set of rules for each part of speech that could be formed into a question word in Telugu. We maximized the possible number of questions in each sentence with all the keywords.

2 Related Work

Previously, Holy Lovenia et al.[2018] experimented on Q&A pair Generation(Holy Lovenia and Gunawan, 2018) in English where they succeeded in forming "What", "Who", and "Where" questions. Rami Reddy et al.[2006] worked on Dialogue based Question Answering System in Telugu for Railway inquiries(Rami Reddy, 2006), which majorly concentrated on Answer Generation for a given Query. Shudipta Sharma et al. worked on implementing automatic Q&A pair generation for English and Bengali texts(Sharma and Hossen, 2018) using NLP tasks like verb decomposition, subject auxiliary inversion for a question tag. Telugu dependency parsing using different statistical parsers (SeshuKumari and RajeshwaraRao, 2017) explored dfferent statistical dependency parsers for parsing Telugu and analysed the performanced of each parser. We explored other[1] Q&A state of art systems from different authors that suita our approach.

[1](Xu J and R., 2004),(Anne R. Diekema, 2004),(Bert Green, 1961),(Hai and KOSSEIM, 2007)

Proceedings of the 17th International Conference on Natural Language Processing, pages 355–361
Patna, India, December 18 - 21, 2020.

3 Summarization

Since Telugu is a low resource language, we used statistical and unsupervised methods for this task[2]. Summarization also ensures the portability of our system to other similar low resource languages.

We have used a Telugu stories dataset taken from a website called "kathalu wordpress".[3] This dataset was chosen because of the variety in the themes of the stories, wide vocabulary and sentences of varying lengths. For summarization, we did the basic data preprocessing (spaces, special characters, etc.) in addition to root-word extraction using Shiva Reddy's POS tagger[4].

We implemented two types of existing summarization techniques:

1. Word Frequency-based summarization
2. TextRank based frequency

3.1 Word Frequency-based Summarization

WFBS (Word Frequency-based Summarization)(Shashikanth and Sanghavi, 2019) is calculated using the word frequency in the passage. This process is based on the idea that the keywords or the main words will frequently appear in the text, and those words with lower frequency have a high probability of being less related to the story.

All the sentences that carry major information are produced successfully by this method because the keywords are used repeatedly in children's stories, subsequently causing the highest frequency.

A ratio is used to get a desirable number of sentences in summary (for example: k% of the sentences). If the first highest frequent word is present k out of 100 sentences, we ratio the word selection to 1:n (where n is the total number of words). This ratio, when dynamically changed, performed better than the fixed ratio of word selection.

Steps followed in WFBS are:

1. Sentences are extracted from the input file
2. Words are preprocessed and tokenized
3. Stop words are removed
4. Frequency of each word is calculated
5. The ratio of words that occur in highest to lowest frequency order is calculated

For testing the meticulousness of the user, as a future task, we can use:

1. The least frequent sentences
2. NE (Named Entities) and CN (Common Nouns) to form questions tags (a next level task)

3.2 TextRank based Frequency

TextRank[5] is a graph-based ranking model that prioritizes each element based on the values in the graph. This process is done in following steps:

1. A graph is constructed using each sentence as a node
2. Similarity between two nodes is marked as the edge weight between nodes
3. Each sentence is ranked based on the similarity with the whole text
4. The page-rank algorithm is run until convergence
5. The sentences with Top N ranking as summarized text is given as the output

The TextRank algorithm is a graph based method that updates the sentence score WS iteratively using the following equation(1).

$$WS(V_i) = (1-d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$

Where d = damping factor (0.85), w_{ij} is the similarity measure between ith and jth sentences. This method has the advantage of using the similarity between the two sentences to rank them instead of high-frequency words. Two kinds of **similarity** measures used:

Common words: A measure of similarity based on the number of common words in two sentences after removing stop words. We used root word extraction of the common words for better results since Telugu is a fusional and agglutinative language and have repeated words with a different suffix each time.

Best Match 25 : A measure of the similarity between two passages, based on term frequencies in the passage.

The results observed by this method capture the crucial information of the story, but lesser readability and fluency are observed. Within the similarity measures, BM25 has shown slightly better results since the BM25 algorithm ranks based on the importance of particular words (inverse document frequency - IDF) instead of just using the frequency of words.

[2](Allahyari and Seyedamin Pouriyeh, 2017)
[3]https://kathalu.wordpress.com/
[4]http://sivareddy.in/downloads
[5](Mihalcea and Tarau, 2004)

4 Answer Phrase Selection

Candidate answers are words/phrases that depict some vital information in a sentence. Adjectives, adverbs, and the subject of a sentence are some examples of such candidates.

The answer selection module utilizes two main NLP components - POS Tagging (Parts Of Speech) and UD parsing (Universal Dependency), along with language-specific rules to determine the answer words in an input sentence.

4.1 POS Tagging

We followed the state of art method called "Cross-Language POS Taggers"(Reddy and Sharoff, 2011) an implementation of a TnT-based Telugu POS Tagger [6] to parse our data.

The tagger learns morphological analysis and POS tags at the same time, and outputs the lemma (root word), POS tag, suffix, gender, number and case marker for each word.

The model was pre-trained on a Telugu corpus containing approximately 3.5 million tokens and had an evaluation accuracy of 90.73% for the main POS Tag.

4.2 UD Tagging

A Bi-LSTM model using Keras is structured and trained using Telugu UD tags dataset "UD_Telugu-MTG". [7]

The Bi-LSTM model outputs the UD Tags for each word in a sentence using Keras. We considered the subject, which is marked "subj" by UD tagger, as a selected answer phrase for a sentence based on a condition that it marked root and punctuation correctly.

This model gave 85% accurate results, including the PAD tags, which might not be an adequate result, but based on the conditions and given that the tags "subj" is labeled in a sentence scarcely, the results have been considered to be acceptable.

4.3 Rules

The outputs of the POS Tagging and UD Parsing modules are used as crucial markers in our language-specific rules. In addition to conditions based on word surroundings, these tags select one or more answer phrases in each sentence.

We classify the rules into different categories, typically based on their usage and interrogative forms.

1. **Quantifiers, Adjectives, Adverbs**: Words with the QC, RB, and JJ POS tag, respectively. For words with JJ tags, the word and corresponding determiners (if present) are selected as the answer candidate.

2. **Possession based**: Words with PRP and NN tags that have suffixes as "టి"," యొక్క", "కి" and "కు" ("ti","yokka","ki" and "ku"). The suffix "టి" ("ti") is used for words like "అతని", "వాళ్ళ", "కంటి", "విద్యార్థుల" ("athani"-his, "valla"-their's, "kanti"-eyes', "vidyarthula"-students')

3. **Time-Place based** : Noun words with a "లో" ("lo") suffix, along with other words present in custom list of time-related words ("మార్నింగ్","ఇయర్")("morning","year") come under this category.

4. **Direct and Reported Speech**: The word "అని" is generally used to denote direct speech in Telugu. Phrases before the word "అని", along with phrases in quotation marks, are chosen as answer phrases.

5. **Verbs**: Telugu follows the SOV(Subject Object Verb) structure for most of its sentences. If the last word has a "V" POS tag, we selected the verb and adjacent adverbs as an answer candidate.

6. **Subject**: We use the UD tags to determine the subject of a sentence. As an additional check, we only select the candidate subjects in those sentences whose last word is tagged as the root verb, and the subject is a noun.

5 Question Formation

Questions are formed according to the chosen phrases chosen previously, and the question words are replaced using further conditions if required.

1. **Quantifiers, Adjectives, Adverbs**: The words that are marked JJ POS are replaced with "ఎటువంటి" ("etuvanti"- what kind of) RB POS tagged that are followed by verbs with "గ" ("ga") suffix are replaced by "ఎలా" ("ela"-how) and the QC tagged words that are not

[6] https://bitbucket.org/sivareddyg/telugu-part-of-speech-tagger/src/master/

[7] https://github.com/UniversalDependencies/UD_Telugu-MTG, (SeshuKumari and RajeshwaraRao, 2017)

articles ("ఒక" (“oka”- one/once)) were chosen and changed based on the following word. If the quantifier is followed by "శాతం","మంది" ,"వరకు" (“shatham”,“mandi”,“varaku”) then the word is replaced with "ఎంత" (“entha”-how much), if the quantifier has a suffix it is added to the question word. For example: "1700కు" - "ఎంతకు" (enthaku) and the rest of the quantifiers are replaced with "ఎన్ని" (“enni”-how many).

2. **Possession based**: The Nouns and Pronouns that satisfied the rules are replaced with "ఎవరి" (“evari”-whose) and the dative cases are replaced with "ఎవరికి" (“evariki”-to whom). This could be an exception for non-animus nouns and pronouns. In the children's stories, most of the nouns are personified, so there were fewer errors than we presumed.

3. **Time-Place based** : We made a list of words that are used to convey time. If the lemma of the word matches the word in the dictionary, then we marked it as “time” and is replaced with "ఎప్పుడు" (“eppudu”-when) or else it is marked as a place and replaced with "ఎక్కడ" (“ekkada”-where).

4. **Direct and Reported Speech** : The whole speech phrase or the phrase that is quoted is replaced with "ఏమని" (“emani”) in the sentence.

5. **Verbs** : The verb is replaced with "ఏమి చేస్తూ" (“emi chesthu”-doing what) + <suffix>”. The appropriate suffix is chosen from the information lost in the lemmatized word.
   Additionally, verb tags were used to form polar questions. The interrogative form of a sentence in Telugu can be constructed by adding intonation to the verb, so we added "ఆ" (“aa”) vowel at the end of the verb to make it a yes or no question. The answer phrase to this question would be "అవును" (“avunu”-yes), followed by the original phrase.

6. **Subject** : Based on the suffix of the verb the subject is replaced with "ఏది", "ఏవి" or "దేని", "వేటికి" (meaning what, which simultaneously) or "ఎవరు" (“evaru”-who) and the root suffix is changed accordingly for "ఎవరు" (“evaru”-who).

Question Word	Occurrences	Errors
ఎలా (ela)	64	2
ఎన్ని (enni)	76	5
ఎంతకు (enthaku)	4	0
ఎంత (entha)	3	0
ఎవరి (evari)	187	0
ఎవరికి (evariki)	1	0
ఏమి (emi)	69	3
దేని (deni)	45	10
ఎవరు (evaru)	20	0
ఎప్పుడు(eppudu)	7	0
ఎక్కడ (ekkada)	21	5
ఏమి చేస్తూ (emi chesthu)	148	2
ఏమని (emani)	10	0
ఆ (aa)	148	0
ఎటువంటి (elanti)	103	6
వేటికి (vetiki)	10	1

Table 1: Question Types.

6 Results

We obtained results that resemble commonly used questions covering nine Parts of Speech. The questions generated by this system are successful and are most similar to questions we see in textbooks. In most cases, it has given legible results that resemble human-made questions, with few exceptions for complex sentences. Out of 916 questions formed, only 34 were either completely erroneous or illegible, the rest of them were both grammatically correct and significant for the context of the story.

Table 1 lists the number of times each question word occurred and the number of times it appeared wrong in the experiment with five short stories.

6.1 Error Analysis

Errors are equally influenced by the word tags, the context of the word, and the word's position in a sentence.

Errors in “ela” ('how') questions are often caused due to spaces between the words and suffixes in the data set we chose.

“enni” (quantifier - based) questions are built from diverse quantifiers (for example: time, age, number of people - these quantifiers are often written as sandhi with the word, which causes the POS tagger to give ambiguous tags) and numerous ways of writing quantifiers in Telugu. Few quan-

tifier question word errors occurred due to wrong POS tagging of cross-coded words (words that are actually in English but written in Telugu script). In Telugu, two numbers are used together when representing non-specific quantities between the two numbers (x y means from x to y), for example, “rendu(two) moodu(three) nimishalu(minutes)” meaning two to three minutes. This kind of representation makes the system assume there are two quantifies, and the sentence is eligible for two questions based on the same.

“deni” (subject-based) questions have errors because of ambiguous suffixes and inaccuracies in UD tagging. The lack of human identification in the system made human subjects also replaceable with “denini” instead of “evarini”. Another error was due to subjects that are names with end syllables similar to common suffixes (which are included as word context in the rule formation). This kind of names were split and formed incorrect question words. The rest of the errors are due to wrong POS tags, cross-codes, and initials/abbreviations.

“emi” (’what’) question forms also have similar POS tags and cross-codes issues. Few of these errors occurred due to punctuation marks between the same sentence breaking it up into multiple sentences.

“etuvanti” (’what-kind-of’) question forms run into issues where there is personification. General questions based on adjectives for humans are based on a person’s subtle qualities; however, in a few cases, the adjective that was chosen is inapt to be formed into a question (less similar to human made question). The question that was formed still is grammatically correct in both human and non-human subjects.

“ekkada” (where) based question forms show errors when an abstract word is used as a place, for example - “In thoughts”, “In that age”. Certain quantitative words in Telugu can be appended with lo - to convey meanings like “in youth”, “in hundreds”. They tend to pass the rules in question generation. Our list of time-related words is not exhaustive, so a few time-related words are also tagged under “ekkada” (place) because of the same suffix.

Most of the tags are error-free except for a few ambiguous errors since the rules select answer phrases precisely or do not consider it.

Some of the examples of the questions that are produced by the system are listed below in Table-2 in the appendix.

The results could be improved to make the question formation precise by increasing the number of rules by observing further data.

The anaphora resolution is a limitation in this system; thus, most of the in-appropriation in the answer section was caused due to this.
For example:
Q: ఎవరి చదువంతా సిటీలో , దర్జాగా ... సాగింది?
Q: Whose studies got completed in the city luxuriously?
A: నీ చదువంతా సిటీలో , దర్జాగా ... సాగింది .
A: Your studies got completed in the city luxuriously.

In this case the question is aptly formed but the answer is slightly ill-formed.

There were few errors due to the POS tagger we used. It marked wrong POS tags for cross coded text becuase of the cross coding and the script differences.

7 Conclusions

We have built a mixed rule-based and AI-based question and answer generating system with 96.28% accuracy.We used two methods for summarization and two similarity measures. We constructed observational-based rules for the data set in a particular domain. There is a chance of varying results if we test this system for data in a different domain, but it gives accuracies above 95% for any data in the domain we chose.

We tested question generation in the news article domain, which gave grammatically correct questions. The error rate may increase if we use complex words and phrases that need tags beyond the proposed set of rules.
We plan to extend our work to be able to include:
1.Anaphora Resolution
2.Extending to other domains
3.Cover more types of questions
4.Increase the accuracy of identifying subject for UD tags

References

Mehdi Allahyari and Saeid Safaei Elizabeth D. Trippe Juan B. Gutierrez Krys Kochut Seyedamin Pouriyeh, Mehdi Assefi. 2017. Text summarization techniques: A brief survey.

Elizabeth D. Liddy Anne R. Diekema, Ozgur Yilmazel. 2004. Evaluation of restricted domain question-answering systems. Proceedings of the Conference on Question Answering in Restricted Domains.

Carol Chomsky Kenneth Laughery Bert Green, Alice Wolf. 1961. An automatic question-answerer.

DOAN-NGUYEN Hai and Leila KOSSEIM. 2007. The problem of precision in restricted-domain question-answering. some proposed methods of improvement.

Felix Limanta Holy Lovenia and Agus Gunawan. 2018. Automatic question-answer pairs generation from text.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing.

Sivaji Bandyopadhyay Rami Reddy, Nandi Reddy. 2006. Dialogue based question answering system in telugu. Proceedings of the Workshop on Multilingual Question Answering - MLQA '06.

Siva Reddy and Serge Sharoff. 2011. Cross language pos taggers for indian languages.

B.Venkata SeshuKumari and Ramisetty RajeshwaraRao. 2017. Telugu dependency parsing using different statistical parsers. *Journal of King Saud University - Computer and Information Sciences*, 29(1):134–140.

Shudipta Sharma and Muhammad Kamal Hossen. 2018. Automatic question and answer generation from bengali and english texts. *Computer Science and Telecommunications 2018*, Volume-54, Issue-2.

Sana Shashikanth and Sriram Sanghavi. 2019. Text summarization techniques survey on telugu and foreign languages. *International Journal of Research in Engineering, Science and Management*, Volume-2, Issue-1.

Licuanan A Xu J and Weischedel R. 2004. Evaluation of an extraction-based approach to answering definitional questions. page 418–424.

A Appendix

Q: ఎటువంటి మోట తో వంగడం కష్టంగా వుంది?
A: అంత పెద్ద మోట తో వంగడం కష్టంగా వుంది

Q: పెప్పు , బట్లు , గాలు , పళ్ళు , గిన్నెలు బజారులో ఎలా కొని , ఊళ్ళో ఇంటింటికి వెళ్లి అమ్ముకునే వాడు?
A: పెప్పులు , బట్టలు , గాజులు , పళ్ళు , గిన్నెలు బజారులో చవకగా కొని , ఊళ్ళో ఇంటింటికి వెళ్లి అమ్ముకునే వాడు

Q: సామాన్లన్నీ మోట కట్టి , గాడిద మీద వేసి , బజారు నుంచి ఊళ్ళో , ఊళ్లోనుంచి తిరిగి ఎవరి ఇంటికి తిప్పేవాడు?
A: సామాన్లన్నీ మోట కట్టి , గాడిద మీద వేసి , బజారు నుంచి ఊళ్ళో , ఊళ్లోనుంచి తిరిగి అతని ఇంటికి తిప్పేవాడు

Q: అమాయక పిచుక ఎక్కడకి, ఎందుకు అని అడగకుండా, ఆ కాకులను గుడ్డిగా నమ్మి ఏమి చేసింది?
A: అమాయక పిచుక ఎక్కడకి, ఎందుకు అని అడగకుండా, ఆ కాకులను గుడ్డిగా నమ్మి వాటితో వెళ్ళింది.

Q: పిచుక మాట నమ్మలేదు కదా , దాని వెపు అసహ్యంగా చూసి మరో ఎన్ని దెబ్బలు వేసారు?
A: పిచుక మాట నమ్మలేదు కదా , దాని వెపు అసహ్యంగా చూసి మరో రెండు దెబ్బలు వేసారు

Q: ఆ కాకులతో పిచుకకి స్నేహం అయ్యిందా?
A: అవును, ఆ కాకులతో పిచుకకి స్నేహం అయ్యింది.

Q: ఒకానొకప్పుడు ఎక్కడ ఒక అమాయకపు పిచుక వుండేది?
A: ఒకానొకప్పుడు ఒక ఊరిలో ఒక అమాయకపు పిచుక వుండేది.

Q: ఏమని పిచుక ప్రాధేయ పడింది?
A: బాబోయ్! బాబోయ్! నా తప్పేమి లేదు, నేను అమాయకురాలిని, నేనేమి చేయలేదు, నన్ను వదిలేయండి! అని పిచుక ప్రాధేయ పడింది.

Table 2: Sample questions generated by the system

List of words related to time:
'అప్పుడు', 'రోజు' , 'కాలం', 'సాయంకాలం', 'ఉదయం', 'మధ్యాహ్నం','రాత్రి','పగలు','నెల','వారం','సంవత్సరం', 'నూర్యాస్తమయం', 'శుభోదయం', 'దినం', 'సమయం', 'వర్తమానం' , 'పూర్వం', 'భవిష్యత్తు', 'సోమవారం', 'మంగళవారం', 'బుధవారం', 'గురువారం', 'శుక్రవారం', 'శనివారం','ఆదివారం','మాసం'

Translations Then, day, time period, evening, morning, afternoon, night, morning(synonym), month, week, year, sunset, sunrise, day(synonym), time, present, past, future, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday, month(synonym).

This set comprises of the time-related words that have a high chance of being used in a storybook.

Table 3: Time Related word list

Q:What kind of sack was hard to carry?
A:That much of a heavy sack was hard to carry.

Q:In the market how was he buying sandals, clothes, bangles, fruits, utensils - and sold them in the village?
A:In the market how was buying sandals, clothes, bangles, fruits, utensils for cheap rates and sold them in the village.

Q:Packing all the things, putting them on the donkey, from market to village, from village to whose house was he taking them?
A:Packing all the things, putting them on the donkey, from market to village, from village to his own house was he taking them.

Q:How did the innocent sparrow believed the crows without even asking why and where?
A:The innocent sparrow believed the crows blindly without even asking why and where.

Q:Instead of believing the sparrow, looking at it with disgust how many times did they beat it?
A:Instead of believing the sparrow, looking at it with disgust they beat it 2 times.

Q:Did the sparrow made friends with the crows?
A:Yes, the sparrow made friends with the crows.

Q:Once upon a time where was the innocent sparrow living?
A:Once upon a time the innocent sparrow was living in a village.

Q:What did the sparrow say pleadingly?
A:The sparrow said ”No! no! i didn’t any mistake, I’m innocent, I did nothing, Please leave me” pleadingly.

Table 4: Translations of the results in Table 2

Detection of Similar Languages and Dialects Using Deep Supervised Autoencoders

Shantipriya Parida[1], **Esaú Villatoro-Tello**[1,2], **Sajit Kumar**[3], **Maël Fabien**[1,4], and **Petr Motlicek**[1]

[1]Idiap Research Institute, Martigny, Switzerland.
{firstname.lastname}@idiap.ch
[2]Universidad Autónoma Metropolitana, Unidad Cuajimalpa, Mexico City, Mexico.
evillatoro@correo.cua.uam.mx
[3]Great Learning, Bangalore, India.
kumar.sajit.sk@gmail.com
[4]Ecole Polytechnique Fédérale de Lausanne, Switzerland
mael.fabien@epfl.ch

Abstract

Language detection is considered a difficult task especially for similar languages, varieties, and dialects. With the growing number of online content in different languages, the need for reliable and robust language detection tools also increased. In this work, we use supervised autoencoders with a bayesian optimizer for language detection and highlight its efficiency in detecting similar languages with dialect variance in comparison to other state-of-the-art techniques. We evaluated our approach on multiple datasets (Ling10, Discriminating between Similar Language (DSL), and Indo-Aryan Language Identification (ILI)). Obtained results demonstrate that SAE is highly effective in detecting languages, up to a 100% accuracy in the Ling10. Similarly, we obtain a competitive performance in identifying similar languages, and dialects, 92%, and 85% for DSL and ILI datasets respectively.

1 Introduction

Internet content is growing exponentially over time, and as a direct consequence, more languages and dialects need to be processed, as they serve as key components in various Natural Language Processing (NLP) tasks (Kocmi and Bojar, 2017).

Language detection is the task of determining the language for a given text. Although language detection has significantly improved over the past years, challenges remain. Detecting similar languages, detecting languages when multiple language contents exist in a single document, and detecting language in short texts are still active research areas (Balazevic et al., 2016; Lui et al., 2014; Williams and Dagli, 2017). Discriminate between very close languages or dialects, for example, German dialect identification, Indo-Aryan language identification, is considered a difficult task (Parida et al., 2020; Jauhiainen et al., 2019a). Although dialect identification is commonly based on the distributions of letters or letter n-grams, these approaches might face serious difficulties when trying to distinguish related dialects that have similar phoneme and grapheme inventories (Scherrer and Rambow, 2010). In a multilingual country like India, there exist many languages and many of them have multiple dialects (Chittaragi and Koolagudi, 2019). For example, in the case of the Odia language, although the written text is the same, there exist many dialects (e.g. Baleswari, Ganjami, Sambalpuri, Desiya. etc.) (Swain et al., 2016). Moreover, the automatic identification of dialect in low resource languages suffers from the lack of large training datasets or pre-trained language models.

Most of the previous research on language identification has focused on using traditional machine learning approaches like Naive Bayes, Support Vector Machine (SVM), in combination with word n-grams, graph-based n-grams, prediction partial matching (PPM) or linear interpolation with post-independent weight optimization and majority voting for combining multiple classifiers (Jauhiainen et al., 2019b). However, more recently, deep learning techniques have shown substantial results in many NLP tasks including language detection (Oro et al., 2018; Villatoro-Tello et al., 2020a,b). For many deep learning tasks, semi-supervised autoencoders have proven to build reliable representations with few annotated data (Ranzato and Szummer, 2008; Rasmus et al., 2015). To the best of our knowledge, autoencoders (AE) have never been applied for similar language detection. In this paper, we explore the use of supervised autoencoders (SAE), hence leveraging labels in the latent space, for language detection.

Proceedings of the 17th International Conference on Natural Language Processing, pages 362–367
Patna, India, December 18 - 21, 2020.

2 Proposed Method

The overall architecture of the proposed method is shown in Figure 1. The following subsections briefly describe the main components of our approach.

2.1 Supervised Autoencoder

An AE is a neural network that learns a low-dimensional representation (encoding) of input data and then learns to reconstruct the original input from the learned representation. This type of architecture is mainly used for dimensionality reduction or feature extraction (Zhu and Zhang, 2019), in an unsupervised fashion. By learning to reconstruct the input, the AE extracts underlying abstract attributes that facilitate accurate prediction of the input.

A supervised autoencoder (SAE) is an AE with the addition of a supervised loss on the representation layer. For the case of a single hidden layer, a supervised loss is added to the output layer and for a deeper AE, the innermost layer has a supervised loss added to the bottleneck layer that is usually transferred to the supervised layer after training the AE.

In supervised learning, the goal is to learn a function for a vector of inputs $\mathbf{x} \in \mathbb{R}^d$ to predict a vector of targets $\mathbf{y} \in \mathbb{R}^m$. Consider SAE with a single hidden layer of size k, and the weights for the first layer are $\mathbf{F} \in \mathbb{R}^{k \times d}$. The function is trained on a finite batch of independent and identically distributed (i.i.d.) data, $(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_t, \mathbf{y}_t)$, with the goal of a more accurate prediction on new samples generated from the same distribution. The weight for the output layer consists of weights $\mathbf{W}_p \in \mathbb{R}^{m \times k}$ to predict $\mathbf{y}$ and $\mathbf{W}_r \in \mathbb{R}^{d \times k}$ to reconstruct $\mathbf{x}$. Let L_p be the supervised loss and L_r be the loss for the reconstruction error. In the case of regression, both losses might be represented by a squared error, resulting in the objective:

$$\frac{1}{t}\sum_{i=1}^{t}\Big[L_p(\mathbf{W}_p\mathbf{F}\mathbf{x}_i, \mathbf{y}_i) + L_r(\mathbf{W}_r\mathbf{F}\mathbf{x}_i, \mathbf{x}_i)\Big] = \frac{1}{2t}\sum_{i=1}^{t}\Big[||\mathbf{W}_p\mathbf{F}\mathbf{x}_i - \mathbf{y}_i||_2^2 + ||\mathbf{W}_r\mathbf{F}\mathbf{x}_i - \mathbf{x}_i||_2^2\Big] \quad (1)$$

The addition of supervised loss to the AE loss function acts as regularizer and results (as shown in equation 1) in the learning of the better representation for the desired task (Le et al., 2018). In summary, an SAE represents a neural network that jointly predicts targets and inputs.

2.2 Bayesian Optimizer

In the case of SAE, there are many hyperparameters related to model construction and optimization. AE training and performance often benefit from hyperparameter tuning to avoid over and underfitting.

Bayesian optimization (BO) is a state-of-the-art hyperparameter optimization algorithm that reached competitive performances on several optimizations benchmarks (Snoek et al., 2012; Bergstra and Bengio, 2012). BO is a technique based on Bayes theorem to direct a search for a global optimization problem that is efficient and effective. It works by building a probabilistic model of the objective function, called the surrogate function, that is then searched efficiently with an acquisition function before candidate samples are chosen for evaluation on the real objective function.

2.3 Textual Features

Character n-grams are fed as an input to the SAE. In comparison to word n-grams, which only capture the identity of a word and its possible neighbors, character n-grams are additionally capable of detecting the morphological makeup of a word (Wei et al., 2009; Kulmizev et al., 2017). The extracted n-gram features are input to the deep SAE as shown in the Figure 1. The deep SAE contains multiple hidden layers. Hyperparameters were optimized using BO.

3 Experimental Setup and Datasets

3.1 Hyperparameters

To verify the robustness of our proposed model, we have used datasets that are either short, contain similar dialects, or cover multiple languages, and long texts. The range of values for the hyperparameters search space is shown in Table 1. During training, BO chooses the best hyperparameters from this range. The overall configuration of the SAE model is shown in Table 2.

3.2 Datasets

A summary table of the number of texts per dataset is presented in Table 3. We also provide a brief description of each dataset.

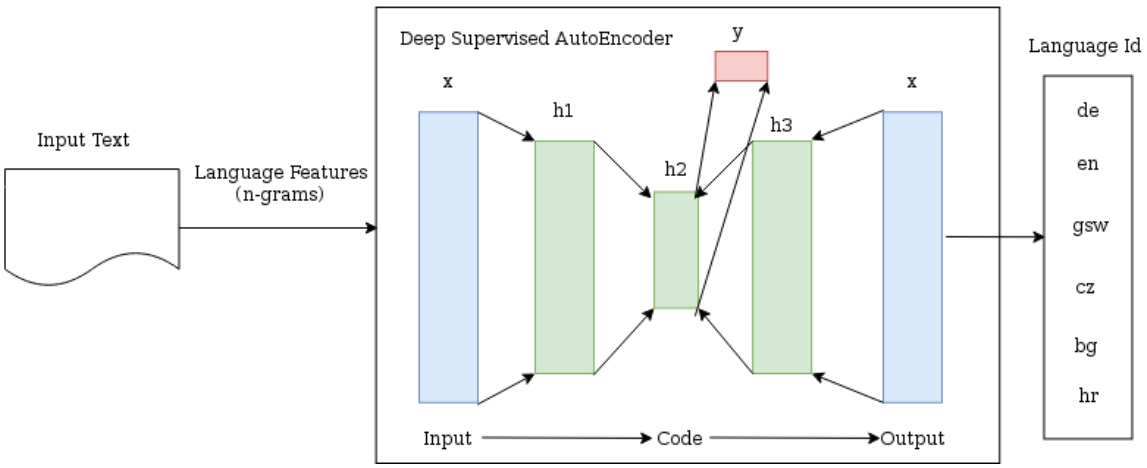


Figure 1: Proposed model architecture. The extracted features of the text are input to the supervised autoencoder. The target "y" are included. The classification output are the language id for the classified languages.

Hyper Parameter	Range
number of layer	1-5
learning rate	$10^{-5} - 10^{-2}$
weight decay	$10^{-6} - 10^{-3}$
activation functions	'relu', 'sigma'

Table 1: Search space hyper parameter range.

Parameter	DSL	Ling10	ILI
`n_gram` range	1-3	1-3	1-3
number of target	14	10	5
embedding dimension	300	300	300
supervision	'clf'	'clf'	'clf'
converge threshold	0.00001	0.00001	0.00001
number of epochs	300	500	500

Table 2: SAE model configurations for the dataset.

Dataset	Training	Development	Test
DSL	252,000	28,000	14,000
Ling10	140,000	-	50,000
ILI	70,351	10,329	9,692

Table 3: Dataset Statistics.

DSL Dataset: The data obtained from the "Discriminating between Similar Language (DSL) Shared Task 2015" contains 13 different languages belonging to 6 language groups, namely South Eastern Slavic (Bulgarian and Macedonian), South Western Slavic (Bosnian, Croatian and Serbian), West-Slavic (Czech and Slovak), Ibero-Romance Spanish (Peninsular Spanish and Argentinian Spanish), Ibero-Romance Portuguese (Brazilian Portuguese and European Portuguese), and Austronesian (Indonesian and Malay). The DSL corpus collection [1] have different versions based on different language groups, representing a benchmark dataset for evaluating language identification systems (Tan et al., 2014a). We used the DSLCCv2.0 [2] to perform our experiments (Tan et al., 2014b). In this version, the training set contains 18,000 sentences for each language and the development set contain 2,000 sentences in each language.

Ling10 Dataset: The Ling10 dataset [3] contains 190,000 sentences categorized into 10 languages (English, French, Portuguese, Chinese Mandarin, Russian, Hebrew, Polish, Japanese, Italian, Dutch) mainly used for language detection and benchmarking natural language processing (NLP) algorithms. It has three variants and we have considered "Ling10-train large" in our experiments.

ILI Dataset: The Indo-Aryan Language Identification (ILI) dataset used for the fifth workshop on NLP for similar languages, varieties and dialects (VarDial) at COLING 2018 [4] (Zampieri et al., 2018b). This task was aimed at identifying 5 closely-related languages of the Indo-Aryan language family – Hindi (also known as Khari Boli), Braj Bhasha, Awadhi, Bhojpuri, and Magahi. Considering Indian geographical location and states,

[1] http://ttg.uni-saarland.de/resources/DSLCC/

[2] https://github.com/Simdiva/DSL-Task/tree/master/data/DSLCC-v2.0

[3] https://github.com/johnolafenwa/Ling10

[4] https://github.com/kmi-linguistics/vardial2018

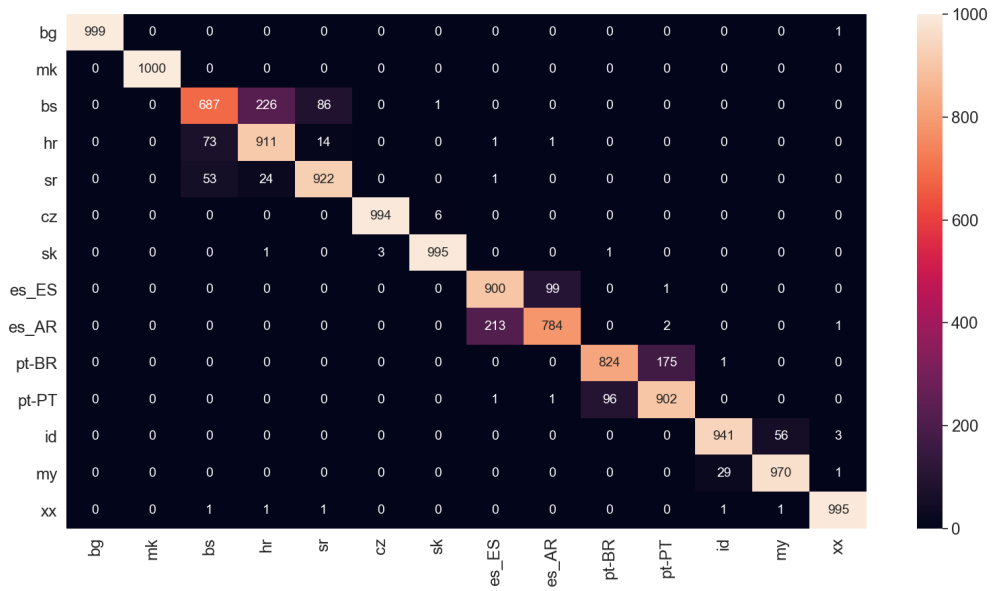


Figure 2: Confusion matrix for the DSL test dataset.

these languages form part of a continuum starting from Western Uttar Pradesh (Hindi and Braj Bhasha) to Eastern Uttar Pradesh (Awadhi and Bhojpuri) and the neighboring Eastern state of Bihar (Bhojpuri and Magahi).

4 Results and Discussion

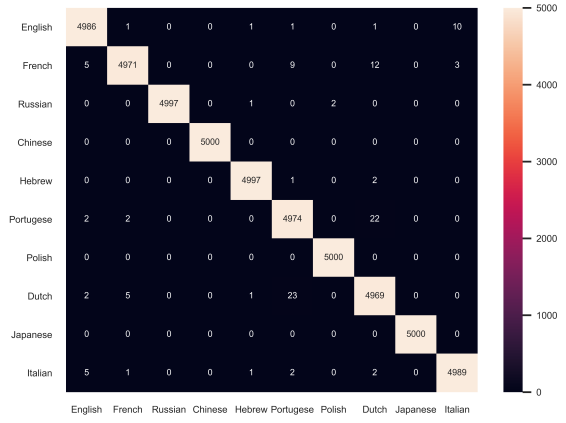


Figure 3: Confusion matrix for Ling10 test dataset.

The SAE model performance for the used dataset is shown in Table 4. Since we are interested in potential confusion between languages, we plot the confusion matrices for the DSL (Figure 2), Ling10 (Figure 3), and for the ILI (Figure 4) datasets.

Observe that for the case of Ling10 (dissimilar

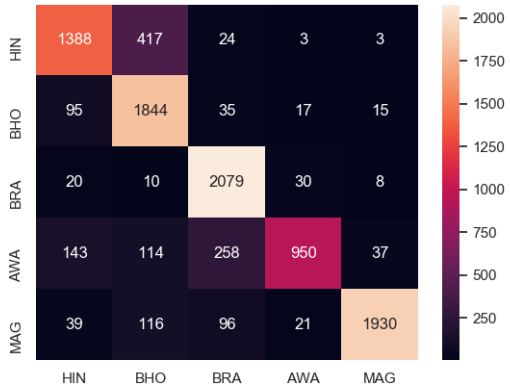


Figure 4: Confusion matrix for ILI test dataset.

language families) the SAE approach performs almost perfect (few confusions between Dutch and Portuguese and Dutch and French) reaching an accuracy of 100%. On the contrary, for DSL and ILI datasets, we can notice more errors. For example, in DSL, there are many mistakes between Spanish, Portuguese, and South Western Slavic families respectively, nevertheless, our SAE gets an accuracy of 92%. Similarly, observe the complexity of the task in the ILI dataset, where the obtained accuracy was of 85%.

As a comparison point, the best-reported result for the DSL dataset is based on a classifier ensemble approach (using 8 SVM classifiers); each one

trained on a single feature type and reached an accuracy of 95.54 % in the test partition during the DSL 2015 shared task using the DSLv2.0 dataset (Malmasi and Dras, 2015; Zampieri et al., 2015). For the case of the ILI dataset, the best score reported during the Second VarDial Evaluation Campaign was of 95% F1-macro (Zampieri et al., 2018a). The winning approach is based on adaptive language models based on character n-grams from 1 to 6 (Jauhiainen et al., 2018). Contrary to these approaches, the proposed SAE represents a much less complex and competitive alternative, obtaining good performance results.

		Accuracy	
Model	Dataset	Validation	Test
SAE (char-3gram)	Ling10	-	100 %
SAE (char-3gram)	DSL	92%	92%
SAE (char-3gram)	ILI	94%	85%

Table 4: Overall performance of the proposed approach.

4.1 Discussion

SAE is less computationally expensive than other deep-learning architectures, while it generalizes well to a wide variety of languages and dialects. The proposed model is extendable by creating a host of features such as character n-gram, word n-gram, word counts, etc, and then passing it through AE to choose the best features. As future work, we are planning to i) verify our model (SAE + BO) with other language detection data sets ii) try to create a dialect detection dataset for other Indian languages and apply SAE for classifying the dialects.

5 Conclusion

In this paper, we introduced SAE with BO for language detection using N-grams at the character level and illustrated its performance on the discrimination of very close languages or dialects on several well-known corpora. We also presented some advantages of the proposed approach, and discuss some of the future directions for SAE-based language detection. [5]

Acknowledgments

This work was supported by the European Union's Horizon 2020 research and innovation program under grant agreement No. 833635 (project ROXANNE: Real-time network, text, and speaker analytics for combating organized crime, 2019-2022). The second author, Esaú Villatoro-Tello, was supported partially by Idiap Research Institute, SNI-CONACyT, CONACyT project grant CB-2015-01-258588, and UAM-C Mexico during the elaboration of this work.

[5] SAE code is available here

References

Ivana Balazevic, Mikio Braun, and Klaus-Robert Müller. 2016. Language detection for short text messages in social media. *arXiv preprint arXiv:1608.08515*.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(Feb):281–305.

Nagaratna B Chittaragi and Shashidhar G Koolagudi. 2019. Automatic dialect identification system for kannada language using single and ensemble svm algorithms. *Language Resources and Evaluation*, pages 1–33.

Tommi Jauhiainen, Heidi Jauhiainen, and Krister Lindén. 2018. Iterative language model adaptation for Indo-Aryan language identification. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, pages 66–75, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Tommi Jauhiainen, Krister Lindén, and Heidi Jauhiainen. 2019a. Language model adaptation for language and dialect identification of text. *Natural Language Engineering*, 25(5):561–583.

Tommi Sakari Jauhiainen, Marco Lui, Marcos Zampieri, Timothy Baldwin, and Krister Lindén. 2019b. Automatic language identification in texts: A survey. *Journal of Artificial Intelligence Research*, 65:675–782.

Tom Kocmi and Ondřej Bojar. 2017. Lanidenn: Multilingual language identification on character window. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 927–936.

Artur Kulmizev, Bo Blankers, Johannes Bjerva, Malvina Nissim, Gertjan van Noord, Barbara Plank, and Martijn Wieling. 2017. The power of character n-grams in native language identification. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 382–389.

Lei Le, Andrew Patterson, and Martha White. 2018. Supervised autoencoders: Improving generalization performance with unsupervised regularizers. In *Advances in Neural Information Processing Systems*, pages 107–117.

Marco Lui, Jey Han Lau, and Timothy Baldwin. 2014. Automatic detection and language identification of multilingual documents. *Transactions of the Association for Computational Linguistics*, 2:27–40.

Shervin Malmasi and Mark Dras. 2015. Language identification using classifier ensembles. In *Proceedings of the joint workshop on language technology for closely related languages, varieties and dialects*, pages 35–43.

Ermelinda Oro, Massimo Ruffolo, and Mostafa Sheikhalishahi. 2018. Language identification of similar languages using recurrent neural networks. In *ICAART*.

Shantipriya Parida, Esaú VILLATORO-TELLO, Sajit Kumar, Petr Motlicek, and Qingran Zhan. 2020. Idiap submission to swiss-german language detection shared task. In *Proceedings of the 5th Swiss Text Analytics Conference (SwissText) & 16th Conference on Natural Language Processing (KONVENS)*, CONF. CEUR Workshop Proceedings.

Marc'Aurelio Ranzato and Martin Szummer. 2008. Semi-supervised learning of compact document representations with deep networks. In *Proceedings of the 25th international conference on Machine learning*, pages 792–799.

Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. 2015. Semi-supervised learning with ladder networks. In *Advances in neural information processing systems*, pages 3546–3554.

Yves Scherrer and Owen Rambow. 2010. Natural language processing for the swiss german dialect area. In *Semantic Approaches in Natural Language Processing-Proceedings of the Conference on Natural Language Processing 2010 (KONVENS)*, pages 93–102. Universaar.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959.

Monorama Swain, Aurobinda Routray, P Kabisatpathy, and Jogendra N Kundu. 2016. Study of prosodic feature extraction for multidialectal odia speech emotion recognition. In *2016 IEEE Region 10 Conference (TENCON)*, pages 1644–1649. IEEE.

Liling Tan, Marcos Zampieri, Nikola Ljubešic, and Jörg Tiedemann. 2014a. Merging comparable data sources for the discrimination of similar languages: The dsl corpus collection. In *Proceedings of the 7th Workshop on Building and Using Comparable Corpora (BUCC)*, pages 11–15.

Liling Tan, Marcos Zampieri, Nikola Ljubešic, and Jörg Tiedemann. 2014b. Merging comparable data sources for the discrimination of similar languages: The dsl corpus collection. In *Proceedings of the 7th Workshop on Building and Using Comparable Corpora (BUCC)*, pages 11–15, Reykjavik, Iceland.

Esaú Villatoro-Tello, Shantipriya Parida, Sajit Kumar, Petr Motlicek, and Qingran Zhan. 2020a. Idiap & uam participation at germeval 2020: Classification and regression of cognitive and motivational style from text. In *Proceedings of the GermEval 2020 Task 1 Workshop in conjunction with the 5th SwissText & 16th KONVENS Joint Conference*, pages 11–16.

Esaú Villatoro-Tello, Gabriela Ramírez-de-la Rosa, Sajit Kumar, Shantipriya Parida, and Petr Motlicek. 2020b. Idiap and uam participation at mex-a3t evaluation campaign. In *Notebook Papers of 2nd SEPLN Workshop on Iberian Languages Evaluation Forum (IberLEF), Malaga, Spain*.

Zhihua Wei, Duoqian Miao, Jean-Hugues Chauchat, Rui Zhao, and Wen Li. 2009. N-grams based feature selection and text representation for chinese text classification. *International Journal of Computational Intelligence Systems*, 2(4):365–374.

Jennifer Williams and Charlie Dagli. 2017. Twitter language identification of similar languages and dialects without ground truth. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 73–83.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Ahmed Ali, Suwon Shon, James Glass, Yves Scherrer, Tanja Samardžić, Nikola Ljubešić, Jörg Tiedemann, Chris van der Lee, Stefan Grondelaers, Nelleke Oostdijk, Dirk Speelman, Antal van den Bosch, Ritesh Kumar, Bornini Lahiri, and Mayank Jain. 2018a. Language identification and morphosyntactic tagging: The second VarDial evaluation campaign. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, pages 1–17, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Marcos Zampieri, Preslav Nakov, Nikola Ljubešić, Jörg Tiedemann, Shervin Malmasi, and Ahmed Ali, editors. 2018b. *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*. Association for Computational Linguistics, Santa Fe, New Mexico, USA.

Marcos Zampieri, Liling Tan, Nikola Ljubešić, Jörg Tiedemann, and Preslav Nakov. 2015. Overview of the dsl shared task 2015. In *Proceedings of the Joint Workshop on Language Technology for Closely Related Languages, Varieties and Dialects*, pages 1–9.

Qiuyu Zhu and Ruixin Zhang. 2019. A classification supervised auto-encoder based on predefined evenly-distributed class centroids. *arXiv preprint arXiv:1902.00220*.

Weak Supervision using Linguistic Knowledge for Information Extraction

Sachin Pawar[1], **Girish K. Palshikar**[1], **Ankita Jain**[1], **Jyoti Bhat**[1] **and Simi Johnson**[2]
[1]TCS Research, Tata Consultancy Services, Pune, India.
[2]Delivery Governance, Risk & Security, Tata Consultancy Services, Chennai, India.
{sachin7.p,gk.palshikar,ankita7.j,jyoti.bhat1,simi.johnson}@tcs.com

Abstract

In this paper, we propose to use linguistic knowledge to automatically augment a small manually annotated corpus to obtain a large annotated corpus for training Information Extraction models. We propose a powerful patterns specification language for specifying linguistic rules for entity extraction. We define an *Enriched Text Format* (ETF) to represent rich linguistic information about a text in the form of XML-like tags. The patterns in our patterns specification language are then matched on the ETF text rather than raw text to extract various entity mentions. We demonstrate how an entity extraction system can be quickly built for a domain-specific entity type for which there are no readily available annotated datasets.

1 Introduction

Much knowledge in an organization resides in text documents of various types. Effectively using the information and knowledge hidden in enterprise document repositories is a challenge. *Information Extraction (IE)* is a well-explored language processing technology for extracting specific kinds of information (e.g., generic or domain-specific entities, relations among entities and events) from documents and presenting it in a structured format (Palshikar, 2012; Pawar et al., 2017; Li et al., 2020). This structured information can then be effectively searched, disseminated, reused or mined using data mining techniques to discover valuable knowledge. IE plays a critical role in several applications such as resumes processing, competitor intelligence from news, patent analysis, and insurance claim management.

IE is posed as a classification task in machine learning, where the training data consists of labeled mentions of a given entity (or relation, or event) type in sentences. Creating a sufficient quantity of such training data is time-consuming and error-prone. Hence, there has been research in distant supervision methods for automating the process of creating training data, often using other knowledge sources such as DBPedia. In this paper, we map each sentence in the given corpus to an *enriched text format (ETF)* by adding syntactic and semantic information to raw text. We then propose an enriched regular expression language to write linguistic knowledge (rules) to extract mentions of entities and relations from the ETF representation of the sentences. Unlike tools like Snorkel (Ratner et al., 2017) or more complex tree regex languages, our pattern language is simpler, more efficient and has novel features to allow linguistic patterns that use context from multiple sentences. We demonstrate the use of linguistic knowledge to automatically create training data and show that the new training data improves accuracy of IE classification models. We demonstrate this methodology on a completely novel application that extracts risk factors from audit reports of software projects.

2 Linguistic Rules

We express linguistic rules in the form of regular expression patterns which are applied on an ETF text in which rich linguistic information is embedded in the form of XML-like tags.

2.1 Enriched Text Format

We use spaCy (Honnibal and Montani, 2017) for processing any input text and the ETF text is generated as follows: (see examples in Table 1)

- `<SENT>` and `</SENT>` tags are added to mark beginning and end of each sentence.
- Each token identified by the word tokenizer is then encapsulated by its corresponding part-of-speech tag. E.g., `<NN>connection </NN>`.
- Each generic named entity identified by spaCy NER is encapsulated by the identified named entity

Proceedings of the 17th International Conference on Natural Language Processing, pages 368–372
Patna, India, December 18 - 21, 2020.

type. E.g., <ORG>Indian Army</ORG>.
• Dependency children of each verb are identified which are related to the verb with key dependency relations such as *nsubj* (nominal subject), *dobj* (direct object), and *nsubjpass* (passive nominal subject). The entire dependency subtree rooted at such children are encapsulated by the tags of the form <DepRel_VERB>. E.g., <dobj_arrested>···</dobj_arrested>
• For each preposition, the tags of the form <prep_PREP_PARENT> are added to encapsulate the entire prepositional phrase modifying a noun or verb parent. E.g., <prep_of_terrorists>···</prep_of_terrorists>
• Complete dependency subtree rooted at each noun and verb is encapsulated using the tags of the form <DNP_NOUN> and <DNP_VERB>, respectively. E.g., <DNP_terrorists> and <DVP_arrested>
• If a noun or a verb is *negated* (i.e., having a child with dependency relation *neg* such as no, not, never), then the complete dependency subtree rooted at that noun or verb is encapsulated using the tags of the form <NEG_NOUN> and <NEG_VERB>.
• In addition, we add tags in the ETF text which indicate presence in gazetteers of multiple types.

2.2 Patterns Specification Language

We have designed a simple patterns specification language for writing linguistic rules for extracting entity mentions. Each pattern has the following important attributes:
• **Pattern ID**: An unique integer identifier.
• **Pattern properties**: Specify whether the pattern is case-sensitive, whether it is to be applied on the ETF text or plain text, and whether it is to be applied sentence-by-sentence or on the entire text.
• **MainRegex**: A valid regular expression pattern containing at least one *named group*[1]. Entity types to be extracted are used as named groups. This pattern is applied against the input text iteratively. If the pattern is matched successfully, then for each named group in the pattern, the text matched for that particular group is extracted as an entity mention of the corresponding entity type.
• **OuterRegex** (optional): If specified, OuterRegex is a valid regular expression with a named group with name *select*. MainRegex (defined above) gets matched only on the part of the input text selected by the *select* group of the OuterRegex. If OuterRegex not specified, then the MainRegex gets matched on the entire input text.

Table 1 shows an example of a linguistic pattern which extracts entity mentions of type Criminal from news articles. This pattern tries to extract names of the criminals by identifying person names within the direct object phrase of the verbs arrested or detained (see the illustration in Table 1).
Facilities in the patterns specification language for more powerful and effective patterns:
• **Embedded variables** can be used in a linguistic pattern (MainRegex or OuterRegex), using the syntax $<<< VAR >>>$ where VAR is defined separately as a regular expression. Such variables are expanded automatically within the pattern to get the final regular expression pattern.
• **Embedded entity types** can be used in a linguistic pattern (MainRegex or OuterRegex, using the syntax [[[ENTITY_TYPE]]]) where these are replaced with actual entity mentions of that entity type extracted in the same text by *earlier* patterns (i.e., patterns with lower PatternID).

3 Linguistic Rules for Weak Supervision

For extracting entity mentions, sequence labelling techniques such as Conditional Random Fields (CRF) and Long Short-term Memory (LSTM) networks are widely used. These supervised techniques need a significant number of annotated sentences for training to achieve desirable extraction accuracy. For any domain-specific entity type, creating a dataset of such annotated sentences involves significant time, manual efforts and cost.

We propose to augment a small training dataset (L) for such entity extraction task with unlabelled data (U) where labels are automatically obtained using linguistic patterns. A large number of additional sentences can be labelled in this way without any extra time and cost. Although, some manual efforts and expertise are needed for designing the linguistic patterns, the efforts are significantly less as compared to manually annotating a large corpus. The patterns are designed in such a way that each pattern is a high-precision pattern. We observed that a sequence labelling model trained using $L \cup U$ achieves better entity extraction performance as compared to a model trained using only L. Also the supervised sequence labelling model does not learn to imitate the linguistic rules exactly because:
Different feature views: The linguistic rules and a supervised sequence labelling model use two different feature views, similar to Co-training. Although

[1] docs.python.org/3/howto/regex.html

Text: Indian Army arrested two terrorists of Al-Badr terror outfit namely Zahid Sheikh and Shareefudin Ahanger in connection with the murder of Danish Manzoor.
ETF: <nsubj_arrested><DVP_arrested><DNP_army><ORG><NNP>Indian </NNP><NNP>Army </NNP></ORG></DNP_army></nsubj_arrested> <VBD>arrested </VBD><dobj_arrested><DNP_terrorists><CARDINAL><CD>two </CD></CARDINAL><NNS>terrorists </NNS><prep_of_terrorists><IN>of </IN><pobj_of><DNP_outfit><CARDINAL><NNP>Al-Badr </NNP></CARDINAL><NN>terror </NN><NN>outfit </NN><RB>namely </RB><appos_outfit><DNP_sheikh><PERSON><NNP>Zahid </NNP><NNP>Sheikh </NNP></PERSON><CC>and </CC><DNP_ahanger><PERSON><NNP>Shareefudin </NNP><NNP>Ahanger </NNP></PERSON></DNP_ahanger></DNP_sheikh></DNP_outfit></DNP_terrorists></appos_outfit></pobj_of></prep_of_terrorists> </dobj_arrested><prep_in_arrested><IN>in </IN><pobj_in><DNP_connection><NN>connection </NN><prep_with_connection><IN>with </IN><pobj_with><DNP_murder><DT>the </DT><NN>murder </NN><prep_of_murder> <IN>of </IN><pobj_of><DNP_manzoor><NORP><NNP>Danish </NNP></NORP><PERSON><NNP>Manzoor </NNP></PERSON> </DNP_manzoor></DNP_murder></DNP_connection></pobj_of></prep_of_murder></pobj_with></prep_with_connection> </pobj_in></prep_in_arrested><.>. </.></DVP_arrested>
Pattern ID: 3; **Pattern properties**: **N** (not case-sensitive), **E** (to be applied on ETF), **S** (to be applied sentence-by-sentence) **MainRegex**: <PERSON>(?P<Criminal>.*?)</PERSON> **OuterRegex**: <dobj_(arrested|detained)>(?P<select>.*?)</dobj_(arrested|detained)>
OuterRegex match for the *select* named group: <DNP_terrorists><CARDINAL><CD>two </CD></CARDINAL><NNS>terrorists </NNS><prep_of_terrorists><IN>of </IN><pobj_of><DNP_outfit><CARDINAL><NNP>Al-Badr </NNP></CARDINAL><NN>terror </NN><NN>outfit </NN><RB>namely </RB><appos_outfit><DNP_sheikh><PERSON><NNP>Zahid </NNP><NNP>Sheikh </NNP></PERSON><CC>and </CC><DNP_ahanger><PERSON><NNP>Shareefudin </NNP><NNP>Ahanger </NNP></PERSON></DNP_ahanger></DNP_sheikh></DNP_outfit></DNP_terrorists></appos_outfit></pobj_of></prep_of_terrorists>
MainRegex matches for the named group *Criminal*: First match: <NNP>Zahid </NNP><NNP>Sheikh </NNP>; Second match: <NNP>Shareefudin </NNP><NNP>Ahanger </NNP>

Table 1: Example of ETF text and a linguistic pattern matched against the ETF text

the feature views are not mutually exclusive, there are major differences. Most of the linguistic rules use dependency parsing information, which is not used by the sequence labelling model.

Multi-sentence patterns: Some of the linguistic patterns have multi-sentence scope, i.e., they use the context information which is outside the sentence from which an entity mention is identified. However, the sequence labeller processes only one sentence at a time, and hence it can not use any context information outside the current sentence. This enables the sequence labeller to learn additional features from the current sentence itself.

4 Related Work

The most relevant line of work for our linguistic patterns is Semgrex (Chambers et al., 2007), TRegex (Levy and Andrew, 2006), and spaCy Rule-based matching[2]. Semgrex and TRegex allow users to write patterns on dependency and constituency trees, respectively. The patterns are based on regular expression matching for nodes (tokens) and various relationships between the nodes. Rule-based matching provided by spaCy allows users to write regular expression patterns for token-level matching but for more complex rules, Python scripting is necessary. Our patterns specification language is specified purely in terms of regular expressions and allows users to write very powerful patterns using facilities such as MainRegex-OuterRegex combination (e.g., PatternID=3 in Table 2) and embedded entity types (e.g., PatternID=4 in Table 2).

A form of indirect supervision is *distant supervision* (Mintz et al., 2009) where a knowledge base is used to automatically create an annotated dataset. Recently, Snorkel framework (Ratner et al., 2017) was proposed to combine multiple weak supervision sources. However, it is not easily adaptable for sequence labelling tasks. Lison et al. (2020) proposed an entity extraction technique using weak supervision from multiple labelling functions such as entity extraction models trained on other domains, gazettes, heuristic functions etc. In our case, for a domain-specific entity like Risk entity (introduced in the next section), labelling functions based on other domains, gazettes or knowledge bases are not feasible. However, two recent approaches (Safranchik et al., 2020; Liang et al., 2020) look promising for our problem setting where linguistic rules are used for weak supervision for entity extraction and we plan to explore them as future work.

5 Application

Weak supervision using linguistic patterns is especially useful for extraction of domain-specific entity types for which obtaining or creating training data is costly. Hence, we demonstrate its effectiveness for extraction of mentions of one

[2] spacy.io/usage/rule-based-matching

Variable definitions (only partial patterns are shown due to space constraints):

```
NEGATIVE_NOUNS:=((un|non)\W*availability|breach(es)?|discrepanc(y|ies)|lack|delays?|slip(pages?)?|over\W*run···
RIGHT_BOUNDARY:=((,|\.|:)[ ]|\b(due|because|hence|which|who|that|based|if|may)[ ]|\bto[ ][^]*<VB[A-Z]?>)
NEGATIVE_VERBS:=(pending|(impact|delay|affect|hinder|hamper|disrupt)(ed|s|ing)?)
POSITIVE_VERBS:=(developed|maintained|installed|completed|followed|complied|tracked|updated|approved|defined···
```

Extraction patterns (above VARIABLEs are included using <<<VARIABLE>>>, the extracted entity mentions are shown in square brackets):

```
-PatternID=1 MainRegex: <DNP_<<<NEGATIVE_NOUNS>>>>(?P<Risk>((?!\bno[ ]).)*?[ ]((?!\bno[ ]).)*?[ ]((?!\bno[ ]).)*?)
(<<<RIGHT_BOUNDARY>>>|</DNP_<<<NEGATIVE_NOUNS>>>>)
    // E.g., Frequent changes in Tech Stack might lead to [delivery slippage].
-PatternID=2 MainRegex: <nsubj_<<<NEGATIVE_VERBS>>>>(?P<Risk>.*?)(<<<RIGHT_BOUNDARY>>>|</nsubj_<<<NEGATIVE_VERBS>>>>)
    // E.g., [Lack of right combination of skills in resources] may impact the timelines of the project delivery.
-PatternID=3 MainRegex: <nsubjpass_<<<POSITIVE_VERBS>>>>(?P<Risk>.*?\b(not|nt|no)[ ].*?\b<<<POSITIVE_VERBS>>>[ ])
 OuterRegex: ^(?P<select>.*?)</NEG_<<<POSITIVE_VERBS>>>>
    // E.g., It was observed that [assessment on data privacy was not completed].
-PatternID=4 MainRegex: <nsubj_leads?>(?P<Risk>.*?)</nsubj_leads?>.*?\bleads?[ ][^ ]*?\bto[ ][^ ]*?[[[Risk]]]
    // E.g., [Frequent changes in Tech Stack] might lead to delivery slippage.
-PatternID=5 (Multi-sentence)
 MainRegex: <DNP_[A-Za-z]+>(?P<Risk>((?!<SENT>|</SENT>|<<<RIGHT_BOUNDARY>>>).)*)</DNP_[A-Za-z]+>
 OuterRegex: \brisks?[ ][^ ]*?:[ ][^ ]*?</SENT>(?P<select>.*?)$
```

Table 2: Representative linguistic patterns for extraction of Risk entity mentions.

such type – Risk. In large IT services organizations, thousands of projects are going on simultaneously. These projects are routinely audited and as a part of this process, auditors also write their opinion about each project as an audit summary. One of the most important piece of information in these audit summaries is potential *Risks* that the project is facing or may face in near future. We define Risk as an entity type which is any undesirable factor which may have an adverse effect on project objectives or outcomes. E.g., `impacting timelines of the project delivery`, `unavailability of skilled resources`. It is important to note that Risk mentions can be not only noun phrases but also verb phrases. Hence, extraction of Risk mentions is challenging compared to the traditional Named Entity Recognition (NER) task where the entity types (such as PERSON or ORG) are mentioned in the form of noun phrases only.

Table 2 shows some linguistic patterns designed for extraction of Risk entity mentions along with examples of sentences and extracted mentions. **PatternID=4** uses the embedded entity type [[[Risk]]] where the final pattern dynamically substitutes Risk extractions by earlier patterns (`delivery slippage` in this case which is already extracted by the first pattern). **PatternID=5** is a multi-sentence pattern which first identifies a list of sentences immediately followed by "`risks:`" (using OuterRegex) and then extracts noun phrases from such sentences as Risk mentions (using MainRegex).

We used a dataset of 3804 audit summaries consisting of 8046 sentences. We manually annotated Risk entity mentions in 700 of these and used 500 as our training set (L) and remaining 200 as our evaluation set. 3104 unlabelled audit summaries (U) were used to augment the manually annotated training set L using the entity mentions identified in U by the linguistic rules. Table 3 shows the overall entity extraction performance on the evaluation set, using CRF and BiLSTM-CRF (Huang et al., 2015) models. It can be observed that the models trained on $L \cup U$ clearly outperform the models trained only on L as well as only rules-based extraction. Consider the sentence: `The SIT and UAT environment is same, this may impact the quality of the deliverables.` Here, only the BiLSTM-CRF model trained using $L \cup U$ was able to extract the Risk mention `The SIT and UAT environment is same` which was neither extracted by the linguistic rules nor by the model trained only on L.

Technique	P	R	F1
Only Linguistic Rules	0.73	0.38	0.50
CRF trained using L	0.60	0.28	0.38
BiLSTM-CRF trained using L	0.41	0.38	0.39
CRF trained using $L \cup U$	0.59	0.37	0.46
BiLSTM-CRF trained using $L \cup U$	0.65	0.54	**0.59**

Table 3: Extraction accuracy for RISK mentions using the evaluation dataset of 200 Audit summaries

6 Conclusions

We proposed a powerful patterns specification language for specifying linguistic rules for entity extraction which are matched against ETF text. The language is also generalizable to encode linguistic knowledge for relation and event extraction. We demonstrated how an entity extraction system can be quickly built for a domain-specific entity type Risk where a small manually annotated dataset is augmented with a large automatically labelled dataset using linguistic knowledge.

References

Nathanael Chambers, Daniel Cer, Trond Grenager, David Hall, Chloe Kiddon, Bill MacCartney, Marie-Catherine De Marneffe, Daniel Ramage, Eric Yeh, and Christopher D Manning. 2007. Learning alignments and leveraging natural logic. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 165–170.

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. https://spacy.io/.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Roger Levy and Galen Andrew. 2006. Tregex and tsurgeon: tools for querying and manipulating tree data structures. In *LREC*, pages 2231–2234. Citeseer.

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*, page 1.

Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. 2020. Bond: Bert-assisted open-domain named entity recognition with distant supervision. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1054–1064.

Pierre Lison, Aliaksandr Hubin, Jeremy Barnes, and Samia Touileb. 2020. Named entity recognition without labelled data: A weak supervision approach. *arXiv preprint arXiv:2004.14723*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011.

G. K. Palshikar. 2012. Techniques for named entity recognition: A survey. In *Collaboration and the Semantic Web: Social Networks, Knowledge Networks and Knowledge Resources*, pages 191–217. IGI Global.

S. Pawar, G.K. Palshikar, and P. Bhattacharyya. 2017. Relation extraction: A survey. In *arXiv:1712.05191*.

Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 11, page 269. NIH Public Access.

Esteban Safranchik, Shiying Luo, Stephen H Bach, Elaheh Raisi, Stephen H Bach, Stephen H Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, et al. 2020. Weakly supervised sequence tagging from noisy rules. In *AAAI*, pages 5570–5578.

Leveraging Alignment and Phonology for low-resource Indic to English Neural Machine Transliteration

Parth Patel[1], Manthan Mehta[2], Pushpak Bhattacharyya[1] and Arjun Atreya[1]
{parthpatel, pb, arjun}@cse.iitb.ac.in, f20170408@pilani.bits-pilani.ac.in
[1]Department of Computer Science & Engineering
Indian Institute of Technology Bombay, Mumbai, India
[2] Department of Computer Science & Information Systems
Birla Institute of Technology & Science, Pilani, India

Abstract

In this paper we present a novel transliteration technique based on Orthographic Syllable *(OS)* segmentation for low-resource Indian languages *(ILs)*. Given that alignment has produced promising results in Statistical Machine Transliteration systems and phonology plays an important role in transliteration, we introduce a new model which uses alignment representation similar to that of IBM model 3 to pre-process the tokenized input sequence and then use pre-trained source and target OS-embeddings for training. We apply our model for transliteration from *ILs* to English and report our accuracy based on Top-1 Exact Match. We also compare our accuracy with a previously proposed Phrase-Based model and report improvements.

1 Introduction

The process of transliteration is defined in Zhang et al. (2012) as *"the conversion of a given name in the source language (a text string in source script) to a name in the target language (another text string in target script), such that the target language name is: (i) phonemically equivalent to the source name, (ii) conforms to the phonology of the target language, and (iii) matches the user intuition of the equivalent of the source language name in the target language, considering the culture and orthographic character usage in the target language".* This definition of transliteration is apt in the context of Machine Translation since it employs transliteration as a subsystem to handle Named Entities (NEs). We are interested in solving the transliteration problem for Indian to English language pairs and in this paper, we demonstrate the use of OS and pre-trained embeddings to overcome the data sparsity problem that arises in low-resource languages.

The structure of the paper is as follows: Section 2 presents the state of the art on machine transliteration. In section 3 and 4, we describe some background and our proposed approach. Then, in section 5, we present our experiments and results. Finally, in section 6, we present our conclusions and in section 7, we express gratitude to our supporters.

2 Related Work

Arbabi et al. (1994) proposed the very first transliteration system for Arabic to English transliteration. In 1998, Knight and Graehl. (1998) proposed a statistical based approach that back transliterates English to Japanese Katakana which was later adopted for Arabic to English back transliteration by Stalls and Knight. (1998). In 2000, three independent research teams proposed English-to-Korean transliteration models. Other series of work on transliteration has focused on character as a unit of transliteration and using Recurrent Neural Networks. Neural network-based system in the 2016 was proposed by Finch et al. (2016) for multiple language pairs. They used Bi-directional LSTMs for good prefix and suffix generation and were able to surpass the state-of-the-art results of previous systems on the same datasets. Kunchukuttan et al. (2018); Le et al. (2019) used standard encoder-decoder architecture (with attention mechanism (Bahdanau et al., 2014)). Until recently, the best-performing solutions were discriminate statistical transliteration methods based on OS-based statistical machine transliteration (Atreya, 2016) for Indian to English language pairs. We focus on applying OS as a transliteration unit on encoder-decoder architecture (with attention) (Luong

Proceedings of the 17th International Conference on Natural Language Processing, pages 373–378
Patna, India, December 18 - 21, 2020.

et al., 2015).

3 Background Knowledge

We first describe the Orthographic syllables, which form the essence of this transliteration module, introducing a new technique for word segmentation, following which we will formulate the probabilistic model for Grapheme-to-Grapheme alignment, explaining the method to position each orthographic syllable in the word.

3.1 Orthographic Syllables

Indic languages possess greater grapheme to phoneme consistency as compared to English(Atreya, 2016). However, the syllable boundary identification for segmentation of an Indic language word into a list of syllables is extremely challenging because of the presence of Schwa (short 'a' vowel preceded by a consonant unless specified otherwise) and diphthongs (sound formed by combination of two vowels) in the syllable unit. In this work, we have used a variant of Syllable as a unit, called Orthographic Syllable which essentially is 'Syllable $-(minus)$ Coda' (See Figure:1). The algorithm used for the OS segmentation is presented in Algorithm 1.

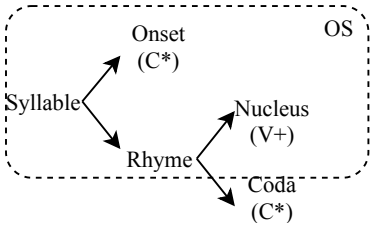


Figure 1: Structure of Orthographic Syllable where the left over Coda concatenates with the Onset of the next OS

3.2 Grapheme-to-Grapheme Alignment

Le et al. (2019) and Yao and Zweig (2015) have reported that, in grapheme-to-phoneme alignments, the grapheme possesses the fertility property similar to IBM Model 3 i.e. it can map to either a null or single or compound phonemes. We assume that same holds true for grapheme-to-grapheme alignments. Since, the aim of alignment is to find a grapheme sequence Y defined by $Y = \{p_1, p_2, ..., p_N\}$, given an OS segmented grapheme sequence X defined by $X = \{o_1, o_2, ..., o_M\}$. Essentially, the problem can be seen as finding the optimal grapheme sequence $\hat{Y}$, which maximizes its conditional probability, as in Equation 1. Since $p(X)$ is independent of the grapheme sequence Y, we can simplify the equation 2 to get Equation 3.

$$\hat{Y} = \arg\max_{Y} p(Y|X) \tag{1}$$

$$\hat{Y} = \arg\max_{Y} \frac{(X|Y)p(Y)}{p(X)} \tag{2}$$

$$\hat{Y} = \arg\max_{Y} p(X|Y)p(Y) \tag{3}$$

Mathematically, given X, Y, and an alignment A, the posterior probability $p(P|O, A)$ is estimated as follows:

$$p(Y|X, A) \approx \prod_{n=1}^{N} p(p_n|p_{n-1}^{n-k}, o_{n-k}^{n+k}) \tag{4}$$

where k is the context window size and n is the alignment position index.
We use expectation-maximization as described in Jiampojamarn et al. (2007) for re-aligning the input sequence after OS segmentation.

Algorithm 1 Orthographic Syllable segmentation . Consonant and Vowel are represented by C and V respectively

```
procedure SEGMENT(word)        ▷ Split word based on regular expression: C*V+
    seg_os ← ""
    prev_vowel ← False
    for each character c in word do
        if prev_vowel and (c!=vowel) then
            prev_vowel ← False
            seg_os ← seg_os + "#"
        end if
        if c = vowel then
            prev_vowel ← True
        end if
        seg_os ← seg_os + c
    end for
    return seg_os.split("#")
end procedure
```

4 Approach

In this section, we present our approach, followed by a description of our experimental

setup, describing the data gathering and cleaning, followed by the model configurations and then the evaluation technique.

4.1 Proposed Approach

Our approach for Indic to English Neural Machine Transliteration consists of 4 steps: *(1) orthographic syllable segmentation, (2) modification of OS-segmented input sequences based on alignment representation, (3) creation of orthographic syllable embeddings with aligned input sequences as input and (4) then we train an RNN-based machine transliteration model.* The whole process is illustrated in Figure 2.

4.2 Experimental Data

We run our experiments on baby names dataset available in multiple *IL* and English from India Child Names website[1] and Bachpan website[2]. The bilingual dataset for learning is divided into training, development, and test-sets at a ratio of 90%, 5% and 5% respectively. The details about the dataset are mentioned in Table 1.

4.3 Model Configuration

We use the *m-2-m aligner*[3] toolkit (Jiampojamarn et al., 2007) to align the training data at OS level. We choose $m = 2$ similar to (Le et al., 2019) for alignment. For the pre-trained source and target OS embeddings, we apply gensims[4] toolkit (Řehůřek and Sojka, 2010) with dimension size of 100, 200, and 300, a window size of 3, and the skip-gram option.

For model training, we apply OpenNMT-py[5] toolkit (Klein et al., 2017) to train our transliteration model. In the transliteration system configuration, we run our model with *Adam* optimizer and use Luong et al. (2015) attention with two learning rates 0.01 and 0.001 for 50000 training steps. We also use 2, 3, and 4 layered encoder-decoder networks(LSTMs) each with vector sizes of 100, 200, and 300 and report the top accuracy values for multiple language pairs.

[1] www.indiachildnames.com
[2] www.bachpan.com
[3] https://github.com/letter-to-phoneme/m2m-aligner
[4] https://radimrehurek.com/gensim/models/fasttext.html
[5] https://github.com/OpenNMT/OpenNMT-py

xx-en pair	train	dev	test	total
Assamese (as)	95308	5295	5295	105897
Bengali (bn)	71832	3991	3991	79813
Gujarati (gu)	16599	923	923	18443
Hindi (hi)	43074	2393	2393	47860
Kannada (kn)	16601	923	923	18445
Malayalam (ml)	15721	874	874	17467
Marathi (mr)	46908	2606	2606	52120
Punjabi (pa)	44737	2486	2486	49707
Tamil (ta)	16393	911	911	18214
Telugu (te)	19970	1110	1110	22188

Table 1: Dataset details for 10 *IL* where xx represents *IL* code mentioned in parenthesis of column 1

4.4 Evaluation Technique

We use Top-1 Exact Match accuracy as the evaluation metric (Banchs et al., 2015). This is one of the metrics used in the NEWS shared tasks on transliteration.

5 Results

We discuss and analyse the results of our experiments, indicating the major improvements and scope of improvement for our approach.

5.1 Results on Test Data

To evaluate our proposed approach, we have implemented three systems (Table 2):

1. Baseline System: We reproduce the results of Atreya (2016) using MOSES[6] toolkit (Koehn et al., 2007) for our experiments along with GIZA++[7] (Och and Ney, 2003) for learning alignments.

2. System 1: Encoder-Decoder LSTM(Klein et al., 2017) + Attention Mechanism(Luong et al., 2015) + OS segmentation(Atreya, 2016)+ one hot encoding as the encoding mechanism.

3. System 2: Encoder-Decoder LSTM(hidden sizes of 128 and 256 were tried) + Attention Mechanism + OS segmentation + pre-trained source and target OS embeddings(sizes of 100, 200 and 300 were used as embedding

[6] https://github.com/moses-smt/mosesdecoder
[7] https://github.com/moses-smt/mgiza

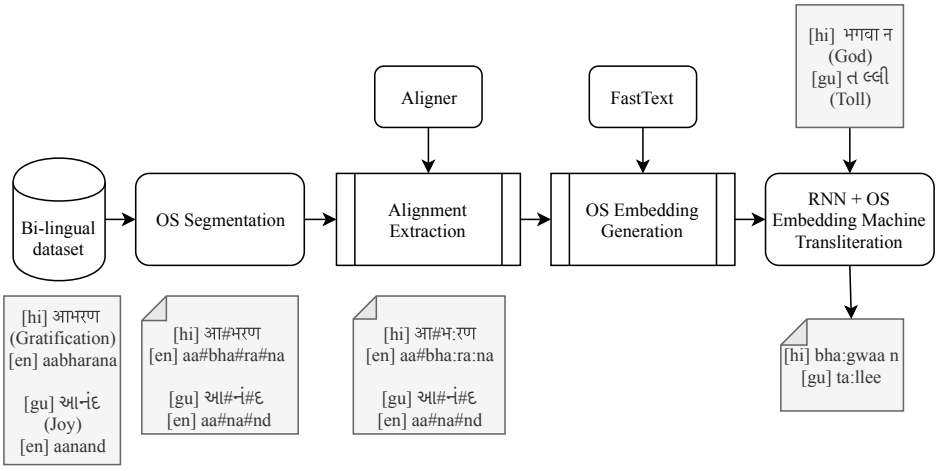


Figure 2: System Architecture for Indic to English Neural Machine Transliteration

Exp.	DS	Indic (xx) to English (en) Language pair									
		as	bn	gu	hi	kn	ml	mr	pa	ta	te
Baseline		50.68	37.75	48.87	57.21	48.94	48.49	45.77	39.28	35.39	46.99
System 1	128	81.92	75.8	73.58	**63.52**	72.7	71.82	67.98	66.51	65.38	69.19
	256	80.43	75.56	73.75	62.96	73.96	71.19	67.43	66.45	65.76	70.22
System 2	128	**82.97**	**78.42**	72.59	62.68	74.29	**74.65**	68.83	**68.73**	65.7	**71.59**
	256	81.54	77.21	**73.84**	62.73	**75.48**	74.54	**69.2**	68.16	**66.77**	69.78

Table 2: Top-1 accuracy figures of xx-en language pairs with 128 and 256 dimension size(DS)

sizes). The learning rate used here was 0.001 with Adam optimizer as already mentioned in section 4.3.

As evident from Table 2, our systems 1 and 2 increase the Top-1 accuracy by at-least **50%**. We list the following observations:

- All language pairs perform better with a learning rate of 0.001 and a 2-layered LSTM.
- We claim that the dimension size is inversely proportional to the size of the dataset for Indic languages. This is supported by the fact that {gu, kn, ta}-en language pairs have smaller dataset size and perform better for LSTM with dimension size of 256. On the other hand, the {as, bn, hi, ml, pa, te}-en language pairs have a larger dataset and perform better with dimension size of 128.

5.2 Error Analysis

Table 3 shows top-10 prediction errors along with actual and predicted output examples.

	y	$\hat{y}$	Count	Expected Word	Output Word
1	ee	i	832	ha mee d	haa mi d
2	aa	a	667	ko maa n	ko ma n
3	i	ee	567	haa mi d	ha mee d
4	th	t	288	vi dva thi	vi dva ti
5	w	v	202	i swa r	i sva r
6	t	th	187	ra nti ka	ra nthi ka
7	a	aa	174	ha mee d	haa mi d
8	v	w	158	vo to n	wo to n
9	c	k	107	mou ni ca	mou ni ka
10	k	c	55	ana mi ka	ana mi ca

Table 3: Top-10 Most confused vowels across all language pairs. y represents the expected output whereas $\hat{y}$ represents the actual predicted output

The most frequent error the system makes is confusing long ई(**E**) sound with a short इ(**e**) and have only predicted correctly 487 times. The characters थ(**th**) and त(**t**), both unaspirated and aspirated consonants, are also mistakenly substituted. *Schwa* present at the end

of an OS also presents a challenge for the prediction since *IL* words are almost always suffixed by a short अ(**a**) sound(unless otherwise specified explicitly by using ◌) that is non-existent in English words. This is also language dependent since राज(raj, rule) from Hindi to English should be transliterated as *Raj* whereas from Dravidian(ta, te, kn, ml) languages should be *Raja*. Similarly, even words of the same language can have two different predictions like मा*(mother)* have *ma* and *maa* which are both correct with respect to English phonology. The characters *w* and *v* are the sounds that both maps to the same akshar of Indo-Aryan languages and are often very difficult to differentiate.

6 Conclusion and Future Work

We show that using pre-trained OS-embeddings on neural encoder-decoder architecture involving OS tokenization outperforms the baseline system by a significant margin. The results also support our claim that phonology and alignment play an important role in increasing the accuracy of transliteration. The reason for the improvement could be learning the Akshar (a combination of vowel and consonant) representation by encoder network and the ability to learn canonical spellings in English.

Given the benefits of using alignment and OS embeddings for low-resource *ILs*, we intend to explore *IL* to *IL* transliteration with and without English as a pivot.

7 Acknowledgement

We would like to show our gratitude to Ministry of Electronics And IT (MEITY[8]) and our colleagues from Center for Indian Language Technology (CFILT[9]) who provided insight, expertise, and resources that greatly assisted the research, and we thank 2 "anonymous" reviewers for comments that improved the manuscript.

References

Mansur Arbabi, Scott M Fischthal, Vincent C Cheng, and Elizabeth Bart. 1994. Algorithms for arabic name transliteration. *IBM Journal of research and Development*, 38(2):183–194.

Arjun Atreya. 2016. Structure cognizant multilingual query expansion in resource scarce languages. Ph.d thesis, IIT Bombay, April.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Rafael E Banchs, Min Zhang, Xiangyu Duan, Haizhou Li, and A Kumaran. 2015. Report of news 2015 machine transliteration shared task. In *Proceedings of the Fifth Named Entity Workshop*, pages 10–23.

Andrew Finch, Lemao Liu, Xiaolin Wang, and Eiichiro Sumita. 2016. Target-bidirectional neural models for machine transliteration. In *Proceedings of the sixth named entity workshop*, pages 78–82.

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 372–379.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational linguistics*, 24(4):599–612.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, pages 177–180.

Anoop Kunchukuttan, Mitesh Khapra, Gurneet Singh, and Pushpak Bhattacharyya. 2018. Leveraging orthographic similarity for multilingual neural transliteration. *Transactions of the Association of Computational Linguistics*, 6:303–316.

Ngoc Tan Le, Fatiha Sadat, Lucie Menard, and Dien Dinh. 2019. Low-resource machine transliteration using recurrent neural networks. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 18(2):13.

[8] https://www.meity.gov.in/
[9] http://www.cfilt.iitb.ac.in/

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

Bonnie Glover Stalls and Kevin Knight. 1998. Translating names and technical terms in arabic text. In *In Proceedings of the Workshop on Computational Approaches to Semitic Languages*, page 34–41.

Kaisheng Yao and Geoffrey Zweig. 2015. Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. *arXiv preprint arXiv:1506.00196*.

Min Zhang, Haizhou Li, Ming Liu, and A Kumaran. 2012. Whitepaper of news 2012 shared task on machine transliteration. In *Proceedings of the 4th Named Entity Workshop*, pages 1–9. Association for Computational Linguistics.

STHAL: Location-mention Identification in Tweets of Indian-context

Kartik Verma[1,*], **Shobhit Sinha**[2,*], **Md Shad Akhtar**[3], **Vikram Goyal**[3]
[1]Delhi Technological University, India
[2]Thapar Institute of Engineering & Technology, India
[3]Indraprastha Instuitute of Information Technology Delhi (IIIT-Delhi), India
vkartik2k@gmail.com, shobhitsinha13@gmail.com
{shad.akhtar, vikram}@iiitd.ac.in

Abstract

We investigate the problem of extracting Indian-locations from a given crowd-sourced textual dataset. The problem of extracting fine-grained Indian-locations has many challenges. One challenge in the task is to collect relevant dataset from the crowd-sourced platforms that contain locations. The second challenge lies in extracting the location entities from the collected data. We provide an in-depth review of the information collection process and our annotation guidelines such that a reliable dataset annotation is guaranteed. We evaluate many recent algorithms and models, including Conditional Random fields (CRF), Bi-LSTM-CNN and BERT (Bidirectional Encoder Representations from Transformers), on our developed dataset named STHAL. The study shows the best F1-score of 72.49% for BERT, followed by Bi-LSTM-CNN and CRF. As a result of our work, we prepare a publicly-available annotated dataset of Indian geolocations that can be used by the research community. Code and dataset are available at https://github.com/vkartik2k/STHAL.

1 Introduction

Location-based systems (Gartner) are playing a vital role in multiple applications such as navigation services, tourist place recommendation, address standardization and safe routes recommendation. To implement such systems and provide location-based services effectively, it requires to have up-to-date information on location names and their associated events. One relevant source for such information is social media which is considered as a crowd-sourced dataset. Social media has become the most potent medium for the real-time source of data (B. Han and Baldwin, 2014) for analysis.

The impact of social media is inevitable and massive. Many case studies reflect on why people are such a lot active on social platforms and share information. This puts the means to connect anywhere, at any time.[1] Journalists also use it as a medium of power to gain insights about the background of various events. However, social networks enable one to harvest recent location events; there lie various challenges due to the platforms being general in terms of sharing of information by individuals. One of the challenges is to select relevant posts out of streaming data that contain location information. The second challenge is to extract location entities from noisy text (Kumar and Singh, 2019) data.

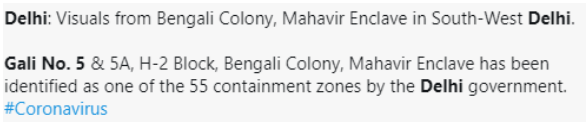


Figure 1: Example of a tweet having non-standard location name

Previous state-of-the-art techniques based on Geolocation Prediction in Twitter (Chi et al., 2016), Detecting Location-Centric Communities (Lim et al., 2015) and Social Media Data Location Prediction (Han et al., 2012) do not perform well in terms of extracting fine-grained location entities in Indian context from the crowd-source data. One of the specific reasons for inferior performance is the non-standardization in location naming conventions. For example, it is easy to find locations names having words such as *gali*, *zila*, *village*, *vihar*, *nagar*, *gaon*, etc. An example is given in Figure 1. It demands designing of specific methods to extract location names and associated events from crowd-source data for Indian subcontinent. The application of this approach can help to demystify the route system, telematics vehicle tracking, and Covid tracking using crowd sourced data.

* First two authors have contributed equally.

[1] https://www.simplilearn.com/real-impact-social-media-article

Proceedings of the 17th International Conference on Natural Language Processing, pages 379–383
Patna, India, December 18 - 21, 2020.

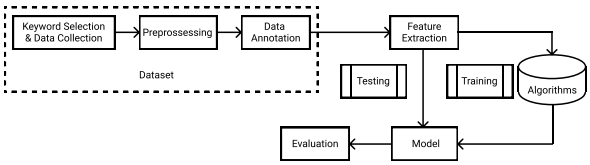


Figure 2: Process flow diagram for the location tagging.

In this paper, we investigate various approaches for named entity recognition focusing mainly on location names. The approaches include Conditional Random Field (CRF) (Lafferty et al., 2001), neural network-based Bi-directional Long Short Term Memory-Convolution Neural Network (Bi-LSTM-CNN) (Chiu and Nichols, 2016), and Bidirectional Encoder Representations from Transformers-based (Devlin et al., 2019) models. We improve the fine-grained location-based approaches by fine-tuning BERT on our annotated dataset (Arase and Tsujii, 2019). We observe a performance improvement of $\geq 2\%$ F1-score points in BERT-based model compared to the other two baseline systems.

The remainder of the paper is organized as follows: In Section 2, we describe the development of STHAL dataset and the proposed approach. We present our experimental results and necessary analyses in Section 3. Finally, we conclude in Section 4.

2 Methodology

In this section, we describe our methodology in detail. First, we explain STHAL's development process, and subsequently, present our system. A high-level diagram is depicted in Figure 2.

2.1 Dataset Development

As discussed above, the need for dataset development is driven by the fact that there exist no[2] tweets dataset to cater the requirements of location-specific entity extraction (or NER, in general) in the Indian context. Therefore, we collect tweet that mentions geographical location (or address) in India and annotates them accordingly.

2.1.1 Data collection

The collection of the dataset was divided into two stages: keyword selection for the seed word list and extraction of tweets for each keyword in the seed list. We adopt the (Aref et al., 2020) method for the collection process.

Keyword Selection In doing the research, the first challenge was to collect comprehensive data for the topic. We mainly focused on collecting the data based on location irrespective of its other attributes, such as the statement's sentiment or the lingual. To tackle this issue, one comprehensive solution can be creating a list of keywords that are concentrated on tweets and crawl using an API.

We created a set of words which was used to extract data from Twitter. We choose *Covid*, *accident*, and *road* as keywords for extraction. These keywords were then iterated with the location database (covered in the next section) to get the required dataset.

Data Set Collection: Among various social media handles, we choose Twitter to extract the dataset because of its wide range of coverage. Twitter is playing an essential role in providing public tweets in the form of JSON document, and it includes various added fields as well such as locations, status Etc.

We use Tweepy[3], a standard Twitter API consist of REST (Representational State Transfer) APIs and Streaming APIs, for data collection. The REST search API provides access to general public tweets with other relevant information. Bulk queries were made to extract the required information. About 250 tweets were only retrieved within the given time frame i.e, 1st January 2020 to 31st August 2020. There was various limitation related to the tweets mentioned in the documentation. Due to these limitations in search API, a custom tweet scraper was made to query the tweets for a given time frame and iterated the keywords and location database (about 233 locations) to get the required database. Around 3500, tweets were retrieved.

2.1.2 Data Pre-processing

Post data collection phase, we apply a series of pre-processing steps to clean our dataset as follows:

- **Removal of irrelevant tweets:** We remove some tweets irrelevant for our case. For example, in the tweet '*Delhi beats Mumbai in Ranji Trophy.*', the mentions of '*Delhi*' and '*Mumbai*' are not referring to a geographical location; instead, they are referring to a cricket team playing for the two cities.

- **Normalization :** In general, tweets consist of lots of noisy texts; therefore, we normalize the

[2]To the best of our knowledge.

[3]https://www.tweepy.org

Text	This	is	the	situation	of	Mahatma	Gandhi	Road	,	Adarsh	Nagar	,	Delhi	33	from	the	last	month	.
Labels	O	O	O	O	O	*B-LOC*	*I-LOC*	*I-LOC*	O	*B-LOC*	*I-LOC*	O	*B-LOC*	*I-LOC*	O	O	O	O	O

Table 1: An example annotated tweet following BIO (Tjong Kim Sang, 2002) scheme from **STHAL dataset**

tweets to remove unprintable, junk, and some special ($, ⋆, Etc.) characters.

Finally, we tokenize the remaining tweets using CMU Ark tokenizer[4] for further processing.

2.1.3 Data Annotation

The annotation process involves the analysis of each tweet in the dataset manual. We adopt BIO notation scheme (Tjong Kim Sang, 2002) to assign a tag (*B-LOC*, *I-LOC*, or *O*) to each token of a tweet. An annotated example is shown in Table 1. The tweet contains three fine-grained locations ('*Mahatama Gandhi Road*', '*Adarsh Nagar*', and '*Delhi 33*') that constitute one coarse-grained location (i.e., '*Mahatama Gandhi Road, Adarsh Nagar, Delhi 33*'). We annotate the location at the fine-grained level. We can observe that the fine-grained locations are separated by punctuation (usually, comma) marks; thus can be easily constructed back to the coarse-grained annotations by assigning *I-LOC* to each intermediate punctuation.

The first token of each location gets a *B-LOC* tags marking the begin of the location. Each subsequent tokens in the location get *I-LOC* tags reflecting the intermediate positions of the location. All non-location tokens are marked with *O* representing outside of the location.

Table 2 lists statistics of the STHAL dataset. In total, it consists of $3,411$ tweets with $8,369$ location mentions.

Stats	**Value**
No. of tweets	$3,411$
No. of tokens	$1,09,162$
No. of locations	8369
Avg. length of sentence	32.002 tokens
Avg. location length	2.255 tokens
Multilingual	English and Romanized Hindi

Table 2: A few statistics of the STHAL dataset.

2.2 Model and Other Baselines

The named-entity-recognition task is a sequence-labelling task, in which, for a sequence of n tokens (i.e., a sentence), we expect a sequence of m tags, where $n == m$. Following the similar setup, we employ BERT (Bidirectional Encoder Representations from Transformer) (Devlin et al., 2019) architecture. To compare the goodness of BERT-based system, we also employ two standard models for sequence labelling task, i.e., a CRF-based model and a Bi-LSTM-CNN (Chiu and Nichols, 2016) architecture.

- **BERT:** BERT as the sequence-learner for the automatic extraction of location mentions from tweets. Recently, BERT has been established as a de facto system for a variety of NLP tasks mainly due to its excellent capability in extracting the underlying semantics from the text. We utilize a pre-trained BERT base model and fine-tune it for the location mention identification in tweets.

- **CRF:** A CRF (Conditional Random Field) (Lafferty et al., 2001) is a class of discriminative model, used for predicting sequences. It exploits the contextual information of the input as well as the predicted labels of the preceding tokens for classifying the current token. The tokens are converted into feature vectors (Quang H Pham, 2019) and are then used by the CRF for sequential labelling. We compute the following three features for the current and previous three tokens: surface form in lower case; a binary feature for all caps; and a binary feature for title case.

- **Bi-LSTM-CNN:** The second system is a pipeline model of Bi-LSTM (Hochreiter and Schmidhuber, 1997) followed by a CNN layer (Kim, 2014). We employ GloVe embeddings (Pennington et al., 2014) model for the feature extraction of input tokens. The hidden representations of Bi-LSTM is fed to a CNN layer and subsequently to the output layer for final classification. To ensure the convoluted features for each token, we zero-padded (Hashemi, 2019) the input. We use 30 trigram filers followed by max-pool (Wu and Gu, 2015) layer.

3 Experiments and Evaluation Results

We implement CRF, Bi-LSTM-CNN, and BERT models in sci-kit-learn, TensorFlow, and PyTorch

[4] http://www.cs.cmu.edu/~ark/TweetNLP/

	Text	Check	distance	from	Gali	No	.	5	,	Dwarka	to	Subzi	Mandi	Old	,	New	Delhi	,	Delhi	-	110036	.
	Gold	O	O	O	*B-LOC*	*I-LOC*	*I-LOC*	*I-LOC*	O	*B-LOC*	*O*	*B-LOC*	*I-LOC*	*I-LOC*	O	*B-LOC*	*I-LOC*	O	*B-LOC*	*I-LOC*	*I-LOC*	O
	CRF	O	O	O	O	O	O	O	O	O	O	*B-LOC*	*I-LOC*	*I-LOC*	O	*B-LOC*	*I-LOC*	O	O	O	O	O
A	**B-CNN**	O	O	O	*B-LOC*	O	O	O	O	O	O	O	O	O	O	*B-LOC*	*I-LOC*	O	*B-LOC*	O	O	O
	BERT	O	O	O	*B-LOC*	*I-LOC*	O	O	O	*B-LOC*	O	*B-LOC*	*I-LOC*	*I-LOC*	O	*B-LOC*	*I-LOC*	O	*B-LOC*	O	*I-LOC*	O
	CRF	O	O	O	*B-LOC*	*I-LOC*	*I-LOC*	*I-LOC*	O	*B-LOC*	O	*B-LOC*	*I-LOC*	*I-LOC*	O	*B-LOC*	*I-LOC*	O	B-LOC	O	O	O
B	**B-CNN**	O	O	O	*B-LOC*	*I-LOC*	*I-LOC*	*I-LOC*	O	*B-LOC*	O	*B-LOC*	*I-LOC*	*I-LOC*	*I-LOC*	*I-LOC*	*I-LOC*	O	*B-LOC*	*I-LOC*	*I-LOC*	O
	BERT	O	O	O	*B-LOC*	*I-LOC*	*I-LOC*	*I-LOC*	O	*B-LOC*	O	*B-LOC*	*I-LOC*	*I-LOC*	O	*B-LOC*	*I-LOC*	O	*B-LOC*	*I-LOC*	*I-LOC*	O

Table 3: A qualitative analysis of the obtained outputs for two setups, A and B. We make two observations: a) Training on dataset with Indian addresses and locations help; and b) BERT yields better outputs (it correctly identifies all five instances of location mentions in setup B) compared to the other two baselines. Red text marks misclassifications.

libraries, respectively. For the evaluation, we utilize CONLL-2002 evaluation script[5] for computing the precision, recall, and F1-score for the location mentions. In all the experiments, we randomly split our annotated dataset, STHAL, into 75:25 ratio for the train and test sets. Moreover, to establish our hypothesis that the exiting NER datasets do not adapt well to the location identifications for the Indian context, we conduct our experiments in two setups.

- **Setup A:** Training on the existing Named Entity Recognition system (NER) [6] dataset and testing on the STHAL's test set.
- **Setup B:** Train and testing on STHAL.

In Table 4, we report our experimental results for both setups on STHAL's test set. All three models, i.e., CRF, Bi-LSTM-CNN, and BERT, yield F1-scores of 19.60%, 26.08%, and 34.03%, respectively, in setup A. One important observation we make here is that the precision of CRF is the highest, while recall is the lowest among all. This suggests that the CRF model is too pessimistic about tagging a token as location-mention, i.e., the low recall value reflects the non-aggressive approach in tagging tokens as location-mentions, and the tokens it tagged as location-mentions are correct 66.23% (precision) of times. The BERT-based model improves upon the recall value but at the cost of low precision; however, the F1-score also improves.

It is evident that the best model in setup A (BERT) does not have good F1-score, mainly due to lack of Indian-styled location-mentions in the train set. In comparison, we observe significant improvements for all models in setup B. The best F1-score of 72.49% is obtained by BERT, followed by Bi-LSTM-CNN (70.31%) and CRF (69.99%).

[5] https://www.clips.uantwerpen.be/conll2002/ner/bin/conlleval.txt

[6] https://www.kaggle.com/abhinavwalia95/entity-annotated-corpus

Setup	Model	Precision	Recall	F1-Score
A	CRF	66.23%	11.50%	19.60%
	Bi-LSTM-CNN	41.26%	19.06%	26.08%
	BERT	34.03%	27.36%	34.03%
B	CRF	75.46%	65.26%	69.99%
	Bi-LSTM-CNN	67.45%	74.41%	70.31%
	BERT	71.98%	73.00%	72.49%

Table 4: Experimental results for location-mention identification on STHAL's test set. It's hard-evaluation, i.e., if any token is misclassified in a location mention, we treat it as the misclassified location mention.

3.1 Error Analysis

We present a qualitative analysis of the obtained outputs in Table 3. For an example tweet in STHAL's test set, we list the token-wise prediction for all three systems in two setups. We make the following two observations:

- In setup A, where the systems are trained on the existing NER dataset (covering global addresses and locations), all systems -including BERT- commit mistakes in identifying *Gali No. 5* as location. In contrast, we observe a better performance of these systems when trained on the STHAL dataset (covering Indian addresses and locations) in setup B.
- In both setups, we observe a superior performance of the BERT-based system compared to the other two baseline systems.

4 Conclusion

In this paper, we present our research on location-mention identification in Indian-context. Due to the lack of representation of Indian-styled location-names and addresses (e.g., *Gali No.*, *chowk*, etc.) in existing datasets, we develop a new Twitter dataset, STHAL, for location-mention identification in Indian context. We benchmark STHALusing BERT-based sequence classifier. Evaluation shows that the underlying system leverages the presence of the Indian-styled location-mentions in train set.

In STHAL, we include location-mentions primarily from Delhi-NCR and northern part of India. Thus, we hypothesis that it may not be adequately sufficient for discovering location-mentions across India, e.g., southern or north-eastern India. In future, we would like to explore the task of location-mentions suitable for the entire nation.

References

Yuki Arase and Junichi Tsujii. 2019. Transfer fine-tuning: A bert case study. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2019)*.

Abdullah Aref, Rana Mahmoud, Khaled Taha, and Mahmoud Al-Sharif. 2020. Hate speech detection of arabic shorttext. In *9th International Conference on Information Technology Convergence and Services (ITCSE 2020)*, pages 81–94.

P. Cook B. Han and T. Baldwin. 2014. Text-based twitter user geolocation prediction. In *Journal of Artificial Intelligence Research*, volume Vol. 49, No. 1, pages 451–500.

Lianhua Chi, Kwan Hui Lim, Nebula Alam, and Christopher J. Butler. 2016. Geolocation prediction in Twitter using location indicative words and textual features. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 227–234, Osaka, Japan. The COLING 2016 Organizing Committee.

P.C. Chiu, Jason and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Krisp Jukka M. Raubal Martin Van de Weghe Nico Gartner, Georg. Location based services: ongoing evolution and research agenda.

Bo Han, Paul Cook, and Timothy Baldwin. 2012. Geolocation prediction in social media data by finding location indicative words. In *Proceedings of COLING 2012*, pages 1045–1062, Mumbai, India. The COLING 2012 Organizing Committee.

Mahdi Hashemi. 2019. Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Abhinav Kumar and Jyoti Prakash Singh. 2019. Location reference identification from tweets during emergencies: A deep learning approach. *International Journal of Disaster Risk Reduction*, 33:365 – 375.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Kwan Hui Lim, Jeffrey Chan, Christopher Leckie, and Shanika Karunasekera. 2015. Detecting location-centric communities using social-spatial links with temporal constraints. In *Advances in Information Retrieval*, pages 489–494, Cham. Springer International Publishing.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Nguyen Viet Cuong Quang H Pham, Binh T Nguyen. 2019. Punctuation prediction for vietnamese texts using conditional random fields. In *SoICT 2019: Proceedings of the Tenth International Symposium on Information and Communication Technology*, page 322–327, New York NY United States. Association for Computational Linguistics.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

Haibing Wu and Xiaodong Gu. 2015. Max-pooling dropout for regularization of convolutional neural networks.

On-Device detection of sentence completion for voice assistants with low-memory footprint

Rahul Kumar Vijeta Gour Chandan Pandey Godawari Sudhakar Rao
Priyadarshini Pai Anmol Bhasin Ranjan Samal
Samsung R&D Institute India, Bangalore
{rahul.k4, vijeta.gour, chandan.p, g.sudhakar, priya.pai,
anmol.bhasin, ranjan.samal}@samsung.com

Abstract

Sentence completion detection (SCD) is an important task for various downstream Natural Language Processing (NLP) based applications. For NLP based applications, which use the Automatic Speech Recognition (ASR) from third parties as a service, SCD is essential to prevent unnecessary processing. Conventional approaches for SCD operate within the confines of sentence boundary detection using language models or sentence end detection using speech and text features. These have limitations in terms of relevant available data for training, performance within the memory and latency constraints, and the generalizability across voice assistant domains. In this paper, we propose a novel sentence completion detection method with low memory footprint for On-Device applications. We explore various sequence-level and sentence-level experiments using state-of-the-art Bi-LSTM and BERT based models for English language.

1 Introduction

Voice-intelligence enabled devices have tremendous potential in providing near natural human behavior based product experience to the users (Dellaert et al. 2020). Such a potential has primarily stemmed from Artificial Intelligence based mimicking of speech recognition, question answering, dialogues, conversations, and command-induced actions. In order to meet user expectations and cater towards product usage satisfaction, knowing when the question, reply or command is complete, i.e., sentence completion detection, is a crucial task.

Voice assistants like Google Assistant, Alexa, Cortana, Siri and Bixby etc. are becoming very popular in modern world. These systems rely on the text, predicted by a streaming speech recognition (ASR) system. Streaming ASR produces text continuously. It is computationally efficient to execute downstream NLP tasks only when a complete sentence is found in the text. This makes SCD a crucial need in voice assistant system. For a partial text received from ASR, the application can wait relatively longer than a complete sentence. Various downstream NLP applications such as unsupervised dependency parsing, ASR transcript readability, accurate information retrieval, and summarization can benefit from SCD.

Detection of sentence completion, has been widely attempted on speech (Hasan, 2014) and text (Azzi, 2019) using sentence boundary detection (SBD) (Sanchez, 2019), end-of-utterance detection (Treviso, 2017), sentence end detection (SED) (Hasan, 2015) models. The techniques evolved from rule-based using handcrafted heuristics (Wang, 2004), to machine learning and more recently, deep learning (Schweter, 2019) based methods. State-of-the-art deep learning architectures reported for SBD include Bi-LSTM CRF (Du, 2019) , BERT (Du, 2019) etc. techniques. On various test datasets, we found their performance highly promising. Upon exploring at further depth, we found that these models have certain limitations with regards to their size, on device platform compatibility and system coupling

Proceedings of the 17th International Conference on Natural Language Processing, pages 384–392
Patna, India, December 18 - 21, 2020.

for on-device deployment. These limitations are briefly described below:

- Size of the models were too big for deployment on memory constrained devices such as mobile phones and smart televisions. MobileBERT[1] takes 100.5Mb with 74ms latency.
- Broad-spectrum conversation data availability, which is representative of a wide range of domains and follows SCD policies as mentioned in Section 4.2
- State-of-the-art architectures reported for SBD lack in ease of modelling with modifications, within the Tensor-flow lite environment.
- Decoupling of SBD models that are a part of bigger system like ASR is challenging and not readily applicable.
- SBD makes use of punctuations and case sensitive information which are missing from immediate ASR output.

We propose that Sentence Completion Detection (SCD) can be achieved by token-level and sentence-level inferencing.

In this research, we explore both token-level and sentence-level inferencing with state-of-the-art language models within on-device deployment constraints. We delve into the tailoring of data, completion detection policies (Section 4.2, SCD Policies), embedding size optimization for achieving a light-weight SCD model that can work on a wide range of domains in memory-constrained environments.

2 Related work

Recent SCD and SBD works are primarily useful for legal text, long lectures, pdf documents etc. Consequently, the datasets used for relevant work included clean texts such as WSJ corpus and Brown Corpus (Francis, 1979), noisy unstructured texts generated from PDFs (Azzi, 2019), (Tian, 2019), lecture (Hasan, 2014) and ASR transcripts (Treviso, 2017), (Rehbein, 2020). For training a model that is suitable for the multi-domain voice assistant, we could not find broad-spectrum domain data focused on commands.

Further, we felt that, essentially, a shift of emphasis from formal, edited text towards more spontaneous language samples which represent ASR output is required Conventional language models are trained on long structured sentences leading to large memory footprints that cannot be supported for fast on-device applications. Various techniques have been reported for downsizing, such as quantization, modifications in vocabulary, truncating input etc.

We expand our work based on modifications in state-of-the-art architectures and extensive custom training with custom loss on multi-domain conversation data. We also experimented upon Tensorflow Lite post training quantization. We primarily looked at Bi-LSTM and BERT architectures as described below.

3 Model

We defined our SCD models in two categories:

- Sequence-based
- Sentence-based

For each of these categories, we explored selected state-of-the-art Bi-LSTM based model and a BERT-based model as described below. Bi-LSTM is a sequence processing model that consists of two LSTMs, one taking the input in a forward direction, and the other in a backwards direction. BERT stands for Bidirectional Encoder Representations from Transformers. It is designed to pre-train deep bidirectional representations from unlabeled text.

3.1 Bi-LSTM and Attention based model for sequence prediction [1.a]

We convert the text input sentence to a sequence of tokens by splitting based on spaces. The model (shown in Figure 1.a) contains an embedding layer which gets trained along with the model and generates vectors for the tokens present in the sentence. For each of the generated tokens feature labelling is done as either '0' or '1' based on the method explained in 4.3. Example token features for a sentence is shown below:

Utterance: "create an event at 5"

Create	An	Event	At	number
0	0	1	0	1

[1]https://www.tensorflow.org/lite/models/bert_qa/overview

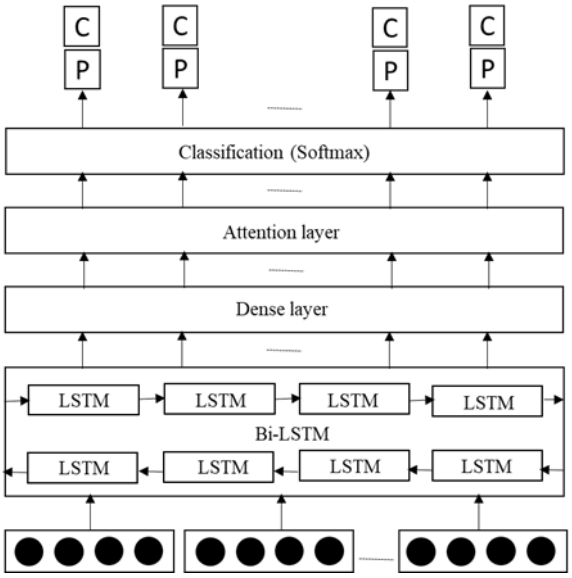


Figure 1.a Bi-LSTM and Attention based model for sequence prediction

Tokens of the input sentence are converted into token IDs. The sentence length used for prediction is kept at 25 tokens, a considerable assumption for sentences in voice assistant based systems. If the sentence is larger than 25 tokens then only the last 25 tokens are used for prediction. So, the resultant input dimensionality becomes 25x1. This resultant vector is then passed to embedding layer which converts it into 25x100 vector followed by a Bi-

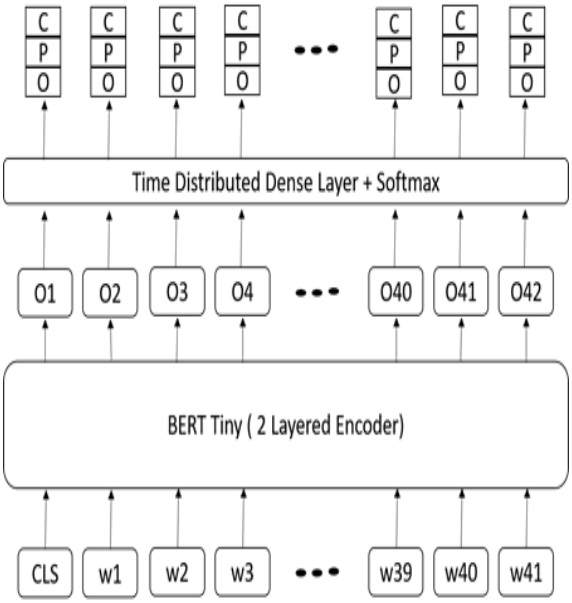


Figure 1.b BERT Tiny based model for sequence prediction

Directional LSTM (Hochreiter, 1997), (Graves, 2005). The output of the Bi-LSTM layer is passed to dense layer in a time distributed manner. The output of dense layer is passed through an attention layer followed by soft-max activation which predicts a label for every token as shown in Figure 1.a. The Loss is calculated using Equation (1) and the inference is done using Equation (2).

3.2 BERT Tiny based model for sequence prediction [1.b]

We convert the text input sentence to a sequence of tokens using Sentence piece tokenizer (Kudo, 2018). For each of the generated tokens feature labeling is done. The tokens generated are marked as '0' or '1' based on method explained in 4.3.
The sequence length used for these models is 42 tokens. So, a sequence of 42 tokens are passed through a pre-trained BERT (Turc, 2019) Tiny model which has got 2 encoder layers with 128 hidden states. The output of the BERT Tiny layer is passed through a dense layer with soft-max activation which predicts a label for every token. The Loss is calculated using the Equation (1) and the inference is done using Equation (2). Figure 1.b below shows the model architecture.

3.3 Bi-LSTM and Attentions based model with hybrid (word + character) embedding for sequence prediction [1.c]

The model 1.a is extended to improve the model performance on sentences containing out of vocab words. To meet that objective we introduced a hybrid embedding strategy. For every token present in the sentence we generate its embedding using char embedding in conjunction with word embedding. Every word is split into characters and then converted into IDs. The maximum length of a word is considered as 10 and first ten characters are taken if the length exceeds the maximum length. The IDs are fed into an embedding layer followed by LSTM sequence. The 50 dimension sequence output of LSTM is concatenated with 100 dimension vector of word embedding. The combined input is passed through a spatial dropout followed by Bi-LSTM. The output of the Bi-LSTM layer is passed to dense layer in a time distributed manner. The output of dense layer is passed through an attention layer followed by soft-max activation which predicts a label for every token. The Loss is calculated using Equation (1) and the

inference is done using Equation (2), which uses soft-max score of the last token.

$$L = l_{partial} + W * l_{complete} \quad (1)$$

Where L is total loss and $l_{partial}$ and $l_{complete}$ are losses on partial and complete tokens in the sentence respectively. $l_{partial}$ is calculated as the sum of categorical cross entropy losses of all the partial tokens. Similarly $l_{complete}$ is calculated as the sum of categorical cross entropy losses of all the partial tokens. W is the ratio of total count of partial tokens to the total count of complete tokens in the training data.

$$Pred = argmax(softmax(Last\_Token)) \quad (2)$$

3.4 Bi-LSTM and Attention based model for sentence classification [2.a]

The input sequence is processed in a similar way as mentioned in Section 3.1. The model contains an embedding layer which gets trained along with the model and generates embedding vectors for the tokens present in the sentence. For every sentence one label is assigned. For partial sentences label '0' is assigned and for complete sentences label '1' is assigned. Each token is converted into IDs. The sentence length used for prediction is kept at 25 tokens. If the sentence is larger than 25 tokens then only the last 25 tokens are used for prediction. So, the resultant input dimensionality becomes 25x1. This resultant vector is then passed to an embedding layer which converts it into 25x100 vector followed by a Bi-Directional LSTM (Hochreiter, 1997), (Graves, 2005). The output of the Bi-LSTM layer is passed to dense layer followed by attention layer followed by soft-max activation which predicts the label for the entire sentence. Figure 2.a shows the model architecture.

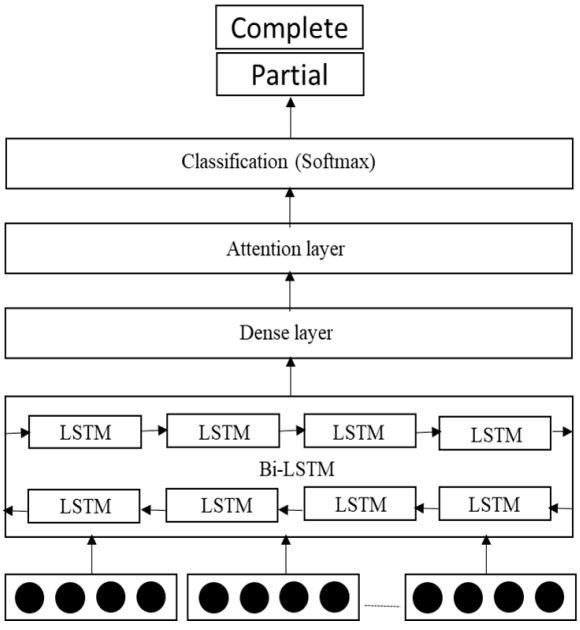


Figure 2.a Bi-LSTM and Attention based model for sentence classification

3.5 BERT based model for sentence classification [2.b]

The sentence tokenization part is similar to the model 1.b. However, this model treats this task as classification. Each sentence is labelled as '1' for complete and '0' for partial. The maximum sequence length used as input is 42 tokens. The tokenized utterance is passed through BERT Tiny model which has got 2 encoder layers with 128 hidden state size. The output of the CLS token of BERT Tiny layer is passed through a dense layer with soft-max activation which predicts a label for the sentence. Figure 2.b shows the model architecture.

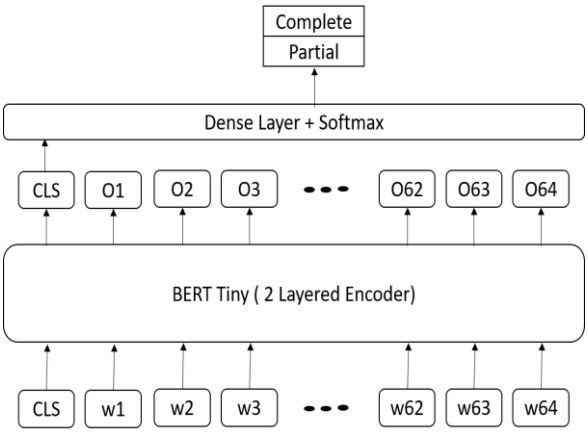


Figure 2.b BERT based model for sentence classification

4 Experimental Setup

4.1 Datasets

For exploring various SCD modelling architectures, we prepared an in-house dataset containing partial and complete sentences representing various domains such as phone call, message, contacts, reminder, maps, accessibility, calculator, clock, open domain, settings, apps, etc. This dataset comprised of sentences of varied

lengths as shown in Table 3. In addition to this, based on our analysis of various available datasets, we selected SNIPS (Coucke, 2018) for gaining insights into model generalizability. The Snips dataset on the other hand is collected from Snips Personal Voice Assistant, spanning 14484 multiple domain utterances. We split these utterances into training, validation and test datasets. From the original utterances present in the dataset we generated partial and complete utterances by generating pre-fixes. We omitted one word pre-fixes from the newly generated utterances.

Dataset	SNIPS	In-House
Train Data	13084	300000
Test Data	700	48000
Validation Data	700	20000
Vocabulary Size	11241	72001
Max Sentence Length	36	89

Table 1. Dataset details

A summary of the training and test data is provided in 3 tables. Table 1 contains the details of the original dataset. Table 2 contains the generated utterances details of SNIPS dataset and Table 3 contains the details of the generated utterances of in-house dataset.

Dataset	SNIPS		
	Total	Partial	Complete
Train	22213	9353	12860
Dev	1296	600	696
Test	1327	628	699

Table 2. Generated utterances details of SNIPS

Dataset	In-House		
	Total	Partial	Complete
Train	2702376	1136745	1565631
Dev	50000	20000	30000
Test	79899	17433	62466

Table 3. Generated utterances details of In-House dataset

4.2 SCD Policies

We aim to make this model highly suited to understand the NLU component in voice assistants for a variety of downstream applications. In each of these aspects, a comprehensive policy formation based on underlying information in relevant data is very important. This, in fact, becomes the key driving factor in determining user experience of the voice assistants. Based on our analysis and understanding, we outlined two main focus areas:

- Intent clarity
- Catchall phrases

We define all those sentences that can elicit an actionable response from the downstream target block in the voice assistant as complete. For example –

"*Create an event*"

Further, sentences that end in open titles are extremely dicey to handle. Any abrupt completion would result in unsatisfactory experience at user's end as there could be multiple complete suffixes for a given sentence. For example –

"*Create a reminder to buy milk*"

In such a scenario, it is difficult to predict if the user intends to continue after "buy milk" with "from a nearby shop". Consequently, strong allocation of sentences with catch-all phrases into partial or complete purely based on semantic understanding will not yield us desirable results. We propose to handle them separately by adopting system-specific suitable behavior.

4.3 Data preprocessing

Before passing a sentence to the model, we preprocessed it .Firstly, we removed punctuations to make the input sentence similar to ASR output. Secondly, we added acronym expansion, and replaced integers with the term "number in order to reduce Out Of Vocabulary (OOV) words. Lastly we removed polite phrases to reduce sentence length. We selected 25 token length for Bilstm models and 42 for BERT models as 98% of the tokenized sentences lengths are covered (Figure 3). This reduces the inference time.

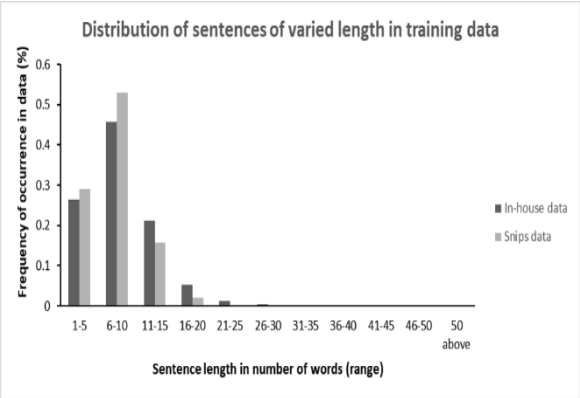


Figure 3 Occurrences of sentences of varied lengths in training data

1. For Bi-LSTM Sequence Models

The tokens present in the sentence were assigned a tag of 0 or 1 depending on whether the intermediate sentence forming up to the current token was already present in the dataset as complete. This was done so as the model sees the sentence as complete even if it is a part of another sentence. If this is not taken care of then the model gets confused for sub-sentences which are complete but also part of a longer sentence. Every intermediate substring of the sentence was checked if it was already present as a complete sentence and the end of the substring was marked as 1.

Utterance: "create a reminder"

create	a	reminder
0	0	1

Utterance: "create a reminder to buy"

create	a	reminder	to	buy
0	0	1	0	0

2. For BERT Sequence Model

The tokens generated by the Sentence Piece Tokenizer were tagged according to strategy mentioned in section 4.3(1). The root token and the subsequent token for the word are both given the same tag.

Utterance: "dont be so judgmental"

don	##t	be	so	judgment	##al
0	0	0	0	1	1

4.4 Training and inferencing

We developed all the models described in section 3 using Tensorflow2.0 as it has a wide collection of workflows, multiple language support and deployment flexibility. We limited the size of the vocabulary of the models- 1.a, 1.c and 2.a to 40000 for in-house dataset and 11000 for SNIPS dataset. BERT tiny based models are initialized with pre trained weights and fine-tuned using recommended hyperparameters (batch size: 32, learning rate: 3e-5, epochs: 3-5) during training.

We selected the best model based on best average F1 score for both the classes of prediction (Partial and Complete) as well as memory foot print.

$$F_1 = \frac{2*Precision*Recall}{Precision+Recall} \quad (3)$$

Once the models were ready, we converted them to TensorFlow Lite with post training quantization for on-device deployment. TensorFlow Lite is designed for efficient model execution on memory constrained devices such as mobile phones. Some of this efficiency reportedly comes from the use of a special storage format that reduces model file size and relevant optimizations that have very less impact on the accuracy.

4.5 Testing

We tested our models on-device on three test datasets as described in 4.1 and report model performances at various inferencing levels, architecture levels and sentence complete detection levels. Further, we also checked the latency and memory footprint to evaluate the feasibility of using such a model on mobile devices.

5 Results and discussion

In the following, we present the results of our performance assessment on various models, *i.e.*, both token-level and sentence-level inferencing using Bi-LSTM and BERT techniques on the three test datasets.

5.1 Performance assessment of various models

Among the BERT and Bi-LSTM models, as shown in Table 4. The best performance is achieved by BERT-Tiny sequence based SCD model. On an average, on both the datasets, it is able to achieve an overall F1-score of 90.95%. The next best

performing model was with Bi-LSTM, attention and word embedding for sentence classification.

Model	SNIPS		In-House Data	
	C	P	C	P
1.a	84.3	77.3	95.5	90.0
1.b	88.0	84.3	96.8	92.3
1.c	82.9	75.2	87.7	88.4
2.a	87.5	83.8	95.69	86.66
2.b	87.8	84.5	93.05	90.19

Table 4. Comparison of F1 scores for partial and complete utterances

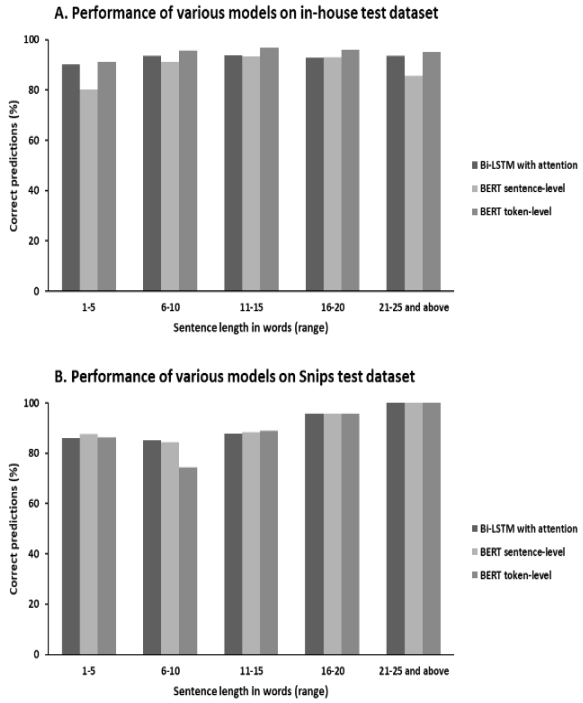


Figure 4. Performance assessment of various models on sentences of different lengths. (A) Results on in-house test data. (B) Results on Snips test data.

Analysis on the results suggests that the sentence completion detection is relatively challenging on very short length (1-5 word) sentences (Figure 4). This might be the reason behind slightly decreased prediction performance (86.87% correct predictions on an average across in-house and Snips test datasets) as compared to sentences with lengths greater than 5 words. We saw most consistent performance on sentence length of 16-20 words at 94.82% average correct predictions across datasets and above 92.62% correct predictions across all model architectures. Possibly a clearer understanding of partial and complete can be achieved by the model in this sentence length range. Further, although there were very few sentences with length beyond 21 words, the models were able to learn completion detection and predict 95.71 % of test data correctly.

5.2 Analytical insights into partial complete sentences prediction

Among the partial and complete sentences tested uisng BERT sequence-based model, we observed that the F1 score for complete sentences was better than partial sentences. We also noticed the same trend in majority of the cases. This is a promising scenario for user experience, where if sentence completion prediction is better on complete sentences, the wait time can be drastically shortened. Consequently, the user experience is also likely to improve.

5.3 Analytical insights into predictions on various test datasets

We observed that in general, the models trained and tested on in-house data performed better than the models trained and tested on SNIPS. The OOV failures were observed less in Word+Char Bi-lstm sequence model and Bert Tiny sequence model.
The sequence models were able to generalize the data better than the sentence models due to the subsequence learning mentioned in Section 4.3.

5.4 Memory footprint of various models

The memory footprint of each of the models developed is given in Table 5. Low memory footprint enables it to be used in memory constrained environment.

Model	Memory (in MB)	
	SNIPS	In-House Data
1.a	6.1	6.8
1.b	4.5	4.5
1.c	4	4.8
2.a	6.1	6.8
2.b	4.5	4.5

Table 5. Comparison of model memory footprint

5.5 On Device Latency of various models

The On Device latency for each of the developed models is mentioned in Table 6. The devices used for testing were Android devices with SDK version 10. The solution works in real time due to low latency.

Model	Latency(in ms)
1.a	22-34
1.b	20-30
1.c	25-37
2.a	15-25
2.b	15-25

Table 6. Comparison of On-device run time latency

6 Conclusion

Sentence completion detection is important for various NLP applications on voice assistant enabled devices. The existing solutions do not cater to the challenges present in conversational ASR output data and are not optimized to work on memory and latency constrained devices. In this paper, we tailored state-of-the-art Bi-LSTM and BERT models for on-device solutions. Fine-tuned BERT Tiny sequence model [1.b] outperforms all other models on both the datasets. Our experimental results show that the mentioned solutions are highly promising for various real-time on-device applications.

References

Du, J., Huang, Y. and Moilanen, K., 2019. IG Investments. AI at the FinSBD Task: Sentence Boundary Detection through Sequence Labelling and BERT Fine-tuning. In *Proceedings of the First Workshop on Financial Technology and Natural Language Processing* (pp. 81-87).

Azzi, A.A., Bouamor, H. and Ferradans, S., 2019. The finsbd-2019 shared task: Sentence boundary detection in pdf noisy text in the financial domain. In *Proceedings of the First Workshop on Financial Technology and Natural Language Processing* (pp. 74-80).

Sanchez, G., 2019, June. Sentence boundary detection in legal text. In *Proceedings of the Natural Legal Language Processing Workshop 2019* (pp. 31-38).

Xu, C., Xie, L. and Xiao, X., 2018. A bidirectional lstm approach with word embeddings for sentence boundary detection. *Journal of Signal Processing Systems*, *90*(7), pp.1063-1075.

Treviso, M.V., Shulby, C.D. and Aluisio, S.M., 2017. Evaluating word embeddings for sentence boundary detection in speech transcripts. *arXiv preprint arXiv:1708.04704*.

Che, X., Luo, S., Yang, H. and Meinel, C., 2016. Sentence Boundary Detection Based on Parallel Lexical and Acoustic Models. In *Interspeech* (pp. 2528-2532).

Ho, T.N., Chong, T.Y. and Chng, E.S., 2016, March. Improving efficiency of sentence boundary detection by feature selection. In *Asian Conference on Intelligent Information and Database Systems* (pp. 594-603). Springer, Berlin, Heidelberg.

Schweter, S. and Ahmed, S., 2019. Deep-EOS: General-Purpose Neural Networks for Sentence Boundary Detection. In *KONVENS*.

Fatima, M. and Mueller, M.C., 2019. HITS-SBD at the FinSBD Task: Machine Learning vs. Rule-based Sentence Boundary Detection. In *Proceedings of the First Workshop on Financial Technology and Natural Language Processing* (pp. 115-121).

Au, W., Chong, B., Azzi, A.A. and Valsamou-Stanislawski, D., 2020, July. FinSBD-2020: The 2nd Shared Task on Sentence Boundary Detection in Unstructured Text in the Financial Domain. In *Proceedings of the Second Workshop on Financial Technology and Natural Language Processing* (pp. 47-54).

Mathew, D. and Guggilla, C., 2019. Ai_blues at finsbd shared task: Crf-based sentence boundary detection in pdf noisy text in the financial domain. In *Proceedings of the First Workshop on Financial Technology and Natural Language Processing* (pp. 130-136).

Tian, K. and Peng, Z.J., 2019. aiai at finsbd task: Sentence boundary detection in noisy texts from financial documents using deep attention model. In *Proceedings of the First Workshop on Financial Technology and Natural Language Processing* (pp. 88-92).

Hirano, M., Sakaji, H., Izumi, K. and Matsushima, H., 2019. mhirano at the finsbd task: Pointwise prediction based on multi-layer perceptron for sentence boundary detection. In *Proceedings of the First Workshop on Financial Technology and Natural Language Processing* (pp. 102-107).

Zhang, R. and Zhang, C., 2020, July. Dynamic Sentence Boundary Detection for Simultaneous Translation. In *Proceedings of the First Workshop on Automatic Simultaneous Translation* (pp. 1-9).

Rehbein, I., Ruppenhofer, J. and Schmidt, T., 2020. Improving sentence boundary detection for spoken language transcripts.

Le, T.A., 2020, January. Sequence Labeling Approach to the Task of Sentence Boundary Detection. In *Proceedings of the 4th International Conference on Machine Learning and Soft Computing* (pp. 144-148).

Wang, D. and Narayanan, S.S., 2004, May. A multi-pass linear fold algorithm for sentence boundary detection using prosodic cues. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing* (Vol. 1, pp. I-525). IEEE.

Oba, T., Hori, T. and Nakamura, A., 2006. Sentence boundary detection using sequential dependency analysis combined with crf-based chunking. In *Ninth International Conference on Spoken Language Processing*.

Hasan, M., Doddipatla, R. and Hain, T., 2014. Multi-pass sentence-end detection of lecture speech. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Hasan, M., Doddipatla, R. and Hain, T., 2015. Noise-matched training of CRF based sentence end detection models. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Dellaert, B.G., Shu, S.B., Arentze, T.A., Baker, T., Diehl, K., Donkers, B., Fast, N.J., Häubl, G., Johnson, H., Karmarkar, U.R. and Oppewal, H., 2020. Consumer decisions with artificially intelligent voice assistants. Marketing Letters, pp.1-13.

Kudo, T. and Richardson, J., 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv: 1808.06226.*

Turc, Iulia and Chang, Ming-Wei and Lee, Kenton and Toutanova, Kristina, 2019. Well-Read Students Learn Better: On the Importance of Pre-training Compact Models. *arXiv preprint arXiv:1908.08962v2*.

Francis, W.N. and Kucera, H., 1979. Brown corpus manual. *Letters to the Editor*, *5*(2), p.7.

Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural computation*, *9*(8), pp.1735-1780.

Graves, A. and Schmidhuber, J., 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural networks*, *18*(5-6), pp.602-610.

Coucke, A., Saade, A., Ball, A., Bluche, T., Caulier, A., Leroy, D., Doumouro, C., Gisselbrecht, T., Caltagirone, F., Lavril, T. and Primet, M., 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Polarization and its Life on Social Media: A Case Study on Sabarimala and Demonetisation

Ashutosh Ranjan
IIIT Hyderabad
ashutosh.ranjan@research.iiit.ac.in

Dipti Sharma
IIIT Hyderabad
dipti@iiit.ac.in

Radhika Krishnan
IIIT Hyderabad
radhika.krishnan@iiit.ac.in

Abstract

This paper is an attempt to study polarisation on social media data. We focus on two hugely controversial and talked about events in the Indian diaspora, namely 1) the Sabarimala Temple (located in Kerala, India) incident which became a nationwide controversy when two women under the age of 50 secretly entered the temple breaking a long standing temple rule that disallowed women of menstruating age (10-50) to enter the temple and 2) the Indian government's move to demonetise all existing 500 and 1000 denomination banknotes, comprising of 86% of the currency in circulation, in November 2016. We gather tweets around these two events in various time periods, preprocess and annotate them with their sentiment polarity and emotional category, and analyse trends to help us understand changing polarity over time around controversial events. The tweets collected are in English, Hindi and code-mixed Hindi-English. Apart from the analysis on the annotated data, we also present the twitter data comprising a total of around 1.5 million tweets.

1 Introduction

Social media, over the past few years, has become the fastest growing medium for expressing opinions for billions of people worldwide. People have always had opinions in history, some extreme and some moderate. What has changed today, is the way people can make their opinions reach masses (read millions) of people in seconds to minutes to hours, depending on the popularity of the person and the general controversiality (for lack of a better word) of the opinion. Earlier people could influence a bunch of people in their vicinity and sometimes, very rarely after going through multiple filters of sanity, they would be able to bring their idea to the world map. Today any random person is able to express their ideas to millions of people, and thereby has the potential to influence masses much faster.

India is no different. Easy access to internet has brought millions of Indians to Twitter, where they express their opinions on topics that interest, affect them. Leaders use social media to make their opinions reach the masses as well. In a recent survey by Pew Research Center it was revealed that a significant percentage of people get their news from social media (Pew Research Center, 2018). This activity on twitter consequently helps people form opinions or change their already existing opinions on different topics. This paper attempts to study this change in opinion of the masses of people by observing and analysing their tweets on social media. We study the life of polarization on social media in the Indian context.

We decide to focus our efforts on two pressing, controversial and divisive issues - 1) Sabarimala 2) Demonetisation. We study how the polarity of opinions of people change with respect to these issues over time. The complete context of the two events is described next.

1.1 Sabarimala

The Sabarimala temple is a temple complex located at Sabarimala inside the Periyar Tiger Reserve in Pathanamthitta district, Kerala, India. It is the site of the one of the largest annual pilgrimages in the world with an estimate of between 17 million and 50 million devotees visiting every year.

According to the Memoir of the Survey of the Travancore and Cochin States, published in two volumes by the Madras government in the 19th century, women of menstruating age were denied entry into the Sabarimala temple two centuries ago, as all sexual intercourse in that vicinity is averse to the celibate deity Lord Ayyappa. In response to a Public Interest Litigation filed in 1991, the Kerala High Court had judged that the restriction of entry of

Proceedings of the 17th International Conference on Natural Language Processing, pages 393–399
Patna, India, December 18 - 21, 2020.

women ages 10-50 to the temple was in accordance with the usage prevalent from time immemorial, and it directed the concerned board to uphold the customary traditions of the temple. However, on 28 September 2018, the Supreme Court of India overturned the restriction on the entry of women, declaring it unconstitutional and discriminatory. On 2nd January 2019, two women under the age of 50 entered the shrine for the first time since the Supreme Court verdict, after attempts of many others failed due to protests by devotees.

The event of two women entering caused huge uproar on social media. One side of the debate argued for the conservation of tradition, and the other side argued for gender equality.

1.2 Demonetisation

In November 2016, the Government of India announced the demonetization of all 500 and 1,000 banknotes of the Mahatma Gandhi Series. It also announced the issuance of new 500 and 2,000 banknotes in exchange for the demonetised banknotes. The Prime minister of India Narendra Modi claimed that the action would curtail the shadow economy and reduce the use of illicit and counterfeit cash to fund illegal activity and terrorism.

The announcement of demonetisation was followed by prolonged cash shortages in the weeks that followed, which created significant disruption throughout the economy. People seeking to exchange their banknotes had to stand in lengthy queues, and several deaths were linked to the rush to exchange cash.

The issue came to relevance again in August 2018 when the Reserve Bank of India released a report, according to which approximately 99.3% of the demonetised banknotes, or 15.30 lakh crore (15.3 trillion) of the 15.41 lakh crore that had been demonetised, were deposited with the banking system, leading analysts to state that the effort had failed to remove black money from the economy. By many economists, the move is blamed for reducing the country's industrial production and slowing down its GDP growth rate.

Initially, the move received support from several bankers as well as from some international commentators. By others it was widely criticised as poorly planned and unfair, and was met with protests, litigation, and strikes against the government in several places across India. The stated failure of demonetisation and subsequent downfall of India's GDP growth rate was also a major topic of interest during the 2019 general elections and cited by many opposition parties as a failure of the presiding government.

These two incidents were divisive and evoked different reactions in people. Both the events have periods where they are focus of attention, and with time the attention dies out. The change in polarity from before the event through the time of the event and then to the eventual end gives us some interesting insights. Here in this study we analyse the change in reactions at the mass level through different time periods, and try to answer the following questions:

- Is society inherently polarised?
- How does a divisive event affect the polarity?

2 Literature Review

Stephen Hawkins and Dixon (2018) published a study entitled "The Hidden tribes of USA". The study, through a survey, categorises the people of USA in 7 major political categories from extreme left to extreme right. They classify four of these seven non-extreme classes as the exhausted majority. The concept of exhausted majority implies that the majority of people lying in the middle of the polarity spectrum are tired from the extremist and tribal stands of the extreme left and extreme right people. They want them to strike a balance in their views and agree on a compromise instead of being engaged in political tribalism. In our work, we see a huge proportion of people keeping neutral opinions which aligns with the above mentioned work.

3 Data Collection

To analyse how people reacted to these two events, we scraped tweets from twitter using keyword search with the help of the api created by Jefferson Henrique (2018). The tweets scraped are collected according to the timeperiods of interest, as mentioned below.

3.1 Sabarimala

We scraped tweets with keyword "sabarimala" for five 3-month periods (Table 1), spread across two years, to study the changing sentiments of people with respect to the event. The 3-month periods are:

- Oct-Dec 2017, a supposedly non controversial period, long before Supreme Court's decision

Time Period	No. of Tweets
Oct - Dec, 2017	6,445
Mar - May, 2018	770
Oct - Dec, 2018	15,373
Jan - Mar, 2019	74,498
May - July, 2019	12,115

Table 1: Sabarimala Twitter Data

Time Period	No. of Tweets
9th Nov'16- 8th Feb'17	711373
May - July, 2017	41461
9th Nov'17 - 8th Feb'18	58464
29th Aug - 29th Nov, 2018	45108
Feb - Apr, 2019	38534

Table 2: Demonetisation Twitter Data

in September 2018. This period also coincides with the days of Mandalapooja festival (around 15 November to 26 December) which is when Sabarimala is the busiest (Dainik Bhaskar, 2011). Hence this period saw a lot of activity on Twitter.

- Mar-May 2018, another supposedly non controversial period. The activity on Twitter was a lot lower during this time with respect to the previous period.
- Oct-Dec 2018, right after Supreme Court Judgement (on 29th September, 2018), but before the date (2nd Jan 2019) the two women entered the temple for the very first time.
- Jan-Mar 2019, right after the two women entered the temple.
- May-July 2019, five to eight months after the controversy.

3.2 Demonetisation

We scraped tweets with keyword "demonetisation" for five 3-month periods (Table 2), spread across two years, to study the changing sentiments of people with respect to the event. The 3-month periods are:

- 8th Nov 2016 - 7th Feb 2017, right after the demonetisation was announced on 8th Nov 2016.
- May-July 2018, 5-8 months after the event.
- 9th Nov 2017 - 8th Feb 2018, 1 year after the announcement.
- 29th Aug 2018 - 29th Nov 2018, right after Reserve Bank of India's review that reported that 99.3% of the cash has been deposited back, effectively stating the failure of demonetisation.
- Feb 2019 - Apr 2019, during general election campaigning days just before the elections in Apr and May 2019.

4 Data Preprocessing

To make preprocessing easier, the tweets were separated into three groups based on the language of content, - 1) English 2) Hindi 3) Hindi-English code-mixed. Tweets containing words from any other language were removed. This separation was done with the help of the language identifier by Bhat et al. (2015).

All three groups of tweets were preprocessed separately as follows:

4.1 Preprocessing English Tweets

- Remove special characters, username mentions (beginning with @) and urls.

4.2 Preprocessing Hindi Tweets

- Remove all special characters, username mentions (beginning with @), and urls.
- Translate the whole tweet to English using python freeware translate.

4.3 Preprocessing Code-Mixed Hinglish Tweets

These tweets were in roman script, with Hindi words written in Roman script as well.

- Use language identifier (Bhat et al., 2015). to find all the Hindi words.
- Use transliterator (Bhat et al., 2015) to transliterate the Hindi words from Roman script to Devanagri script.
- Translate Hindi words in Devanagri script to English using python freeware translate.

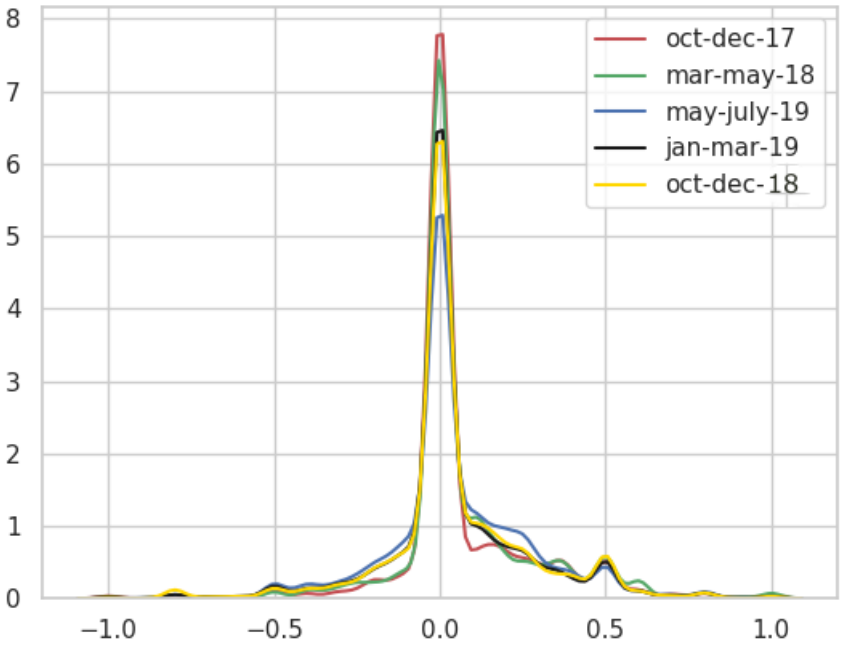


Figure 1: Sabarimala Gaussian Curves measuring polarity percentage

5 Methodology

5.1 Sentiment Analysis

We use the python library TextBlob (Loria, 2018) which uses naive bayes to find the sentiment of each of the tweets on a scale of -1 to 1. We say that the polarity of a tweet increases as we move away from 0 to either side.

5.2 Emotional change through the different periods

We used the NRC Emotion Lexicon (Mohammad and Turney, 2013) to find the emotions expressed and their change through the different periods (Figures 2 and 5). The NRC Emotion Lexicon is a labelled dataset that associates words with emotions, which can be of eight types - anger, sadness, disgust, joy, surprise, trust, anticipation and fear. As a bonus, it also associates each word with either positive or negative connotation.

6 Experiments and Results

6.1 Sabarimala

- Overall negative sentiment increases after the Jan 1 incident. (Figure 1)
- Opinions polarized due to the event (on both sides), in comparison to a non-event (non-controversial) normal state. The graph shows a significant decrease in the neutral population in the subsequent periods. (Figure 1)
- 5-8 months after the polarizing event, the opinions are still polarised and even more so than they were at the time of the event. (Figure 1)
- The number of tweets increase drastically during the period in which the event took place wrt. previous non controversial periods. It also decreases drastically in the period 5-8 months after the event. (Figure 3)
- Grouped emotion bars show increased anger, sadness, negativity. (Figure 2)

6.2 Demonetisation

- The tweets in the first period, i.e. right after when the event happens, are huge in number and decrease drastically in the subsequent periods. (Figure 6)
- The graph shows a significant decrease in the neutral population in the subsequent periods. (Figure 4)

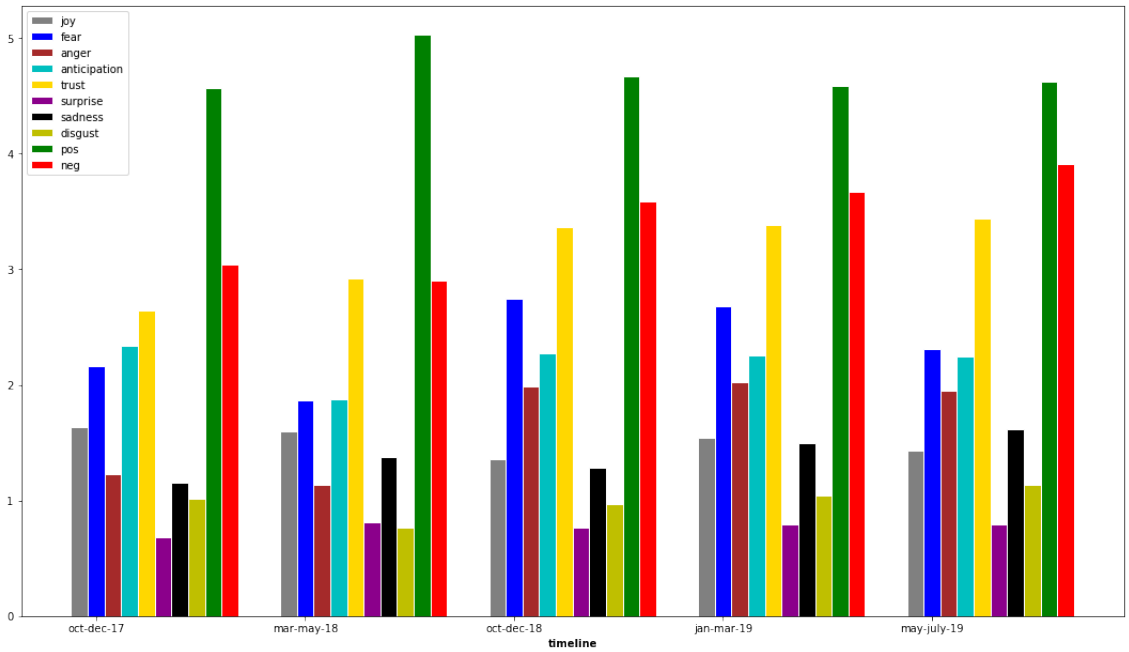


Figure 2: Emotion category Sabarimala

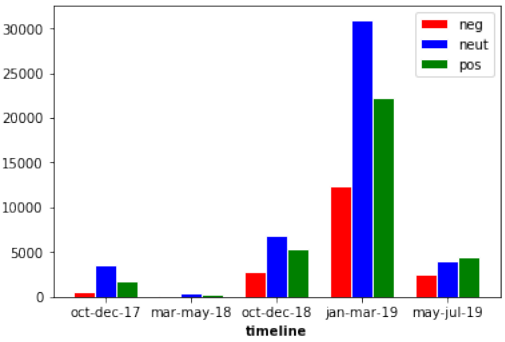


Figure 3: Sabarimala tweet frequency

- The graph shows an increase in percentage of tweets towards negative and positive sides, signalling increase in percentage of polarized reactions. (Figure 4)
- Negativity, sadness emotions have increased through the timelines with the maximum negativity and sadness being in the election period. (Figure 5)

7 Key Observations

- Our study confirms the ”exhausted majority” observation of the Hidden Tribes study. Majority of people have neutral opinion. Can be observed both in Sabarimala and Demonetisation.
- The activity at the time of the event is a lot more than at other times.
- Polarity before the event is negligible.
- Once an event has happened the polarity remains even after a significant amount of time.
- Since 1) the number of people have decreased significantly, 2) percentage of neutral people have decreased significantly 3) People with higher polarity have increased. 1, 2 and 3 make us believe that people with strong opinions remain long after the event while people with not so strong opinions disappear as shown in the analysis of the above two events.

8 Limitations

- Limitations of translation and transliteration.
  - Transliterating Hindi to English transliterate only the words and leaves the sentence structure same.
  - Translation changes the words and the negativity of the words may not be captured in the translated words.
- Limitations of sentiment analyser.
  - The sentiment analyser uses naive bayes and ends up focussing on the toxicity of specific words, rather than getting the sentiment of the entire sentence.

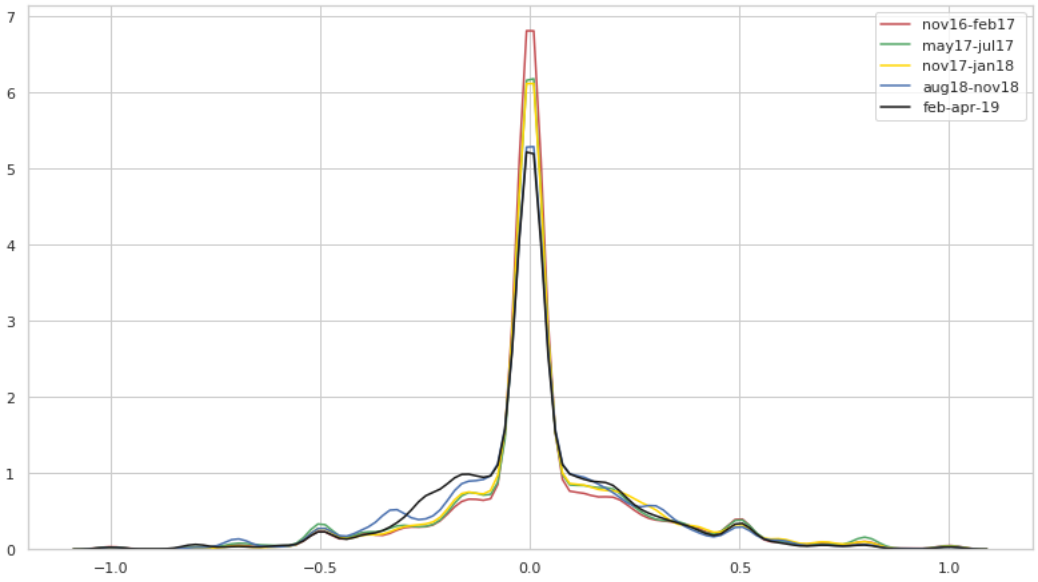


Figure 4: Demonetisation Gaussian curves

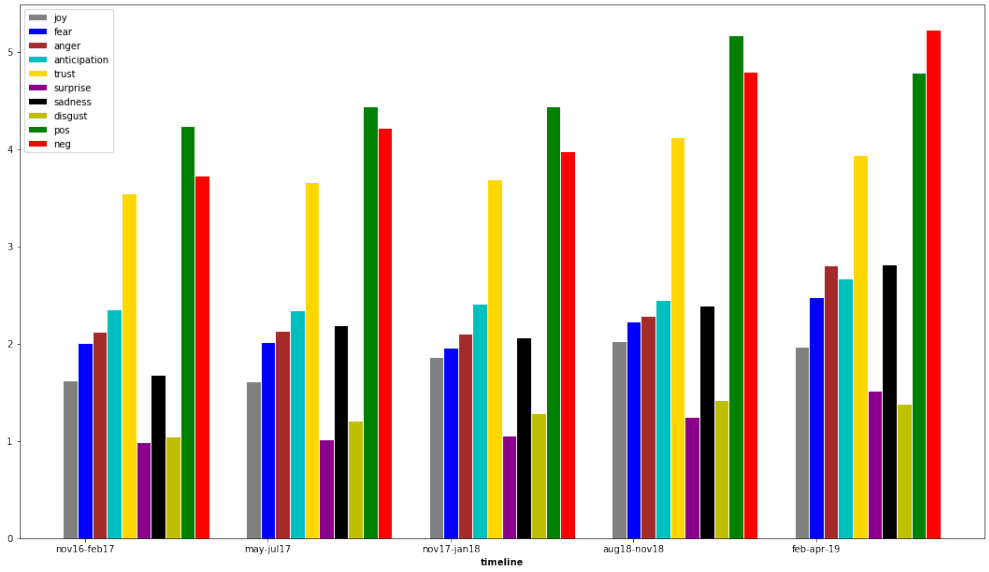


Figure 5: Emotion category Demonetisation

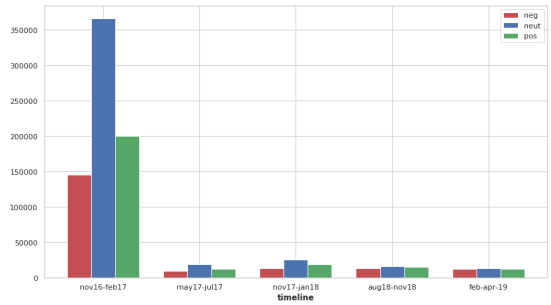


Figure 6: Demonetisation Tweet frequency

- Limitations of emotional category finding tool.
  - Each word is associated with one or more of the eight emotions. The analysis counts the total number of words with each emotion over the whole corpus.
  - Ideally we would want an emotion to be associated with each tweet.
  - Hence even though the overall emotion of the whole corpus is captured by word-by-word emotion categorisation, tweet-by-tweet emotion categorisation might help us gauge the emotions expressed

better.

9 Conclusion

We realize that:

- Majority of people hold neutral opinions.
- The polar opinions that seemingly remain after months are because of the people who were polar even before.

References

Irshad Ahmad Bhat, Vandan Mujadia, Aniruddha Tammewar, Riyaz Ahmad Bhat, and Manish Shrivastava. 2015. Iiit-h system submission for fire2014 shared task on transliterated search. In *Proceedings of the Forum for Information Retrieval Evaluation*, FIRE '14, pages 48–53, New York, NY, USA. ACM.

Dainik Bhaskar. 2011. Why millions throng sabarimala shrine.

Jefferson Henrique. 2018. Getoldtweets api.

Steven Loria. 2018. Textblob.

Saif Mohammad and Peter Turney. 2013. Crowdsourcing a word-emotion association lexicon. *Computational Intelligence*, 29.

Pew Research Center. 2018. Americans are wary of the role social media sites play in delivering the news.

Míriam Juan-Torres Stephen Hawkins, Daniel Yudkins and Tim Dixon. 2018. The hidden tribes of usa. In *More in Common*.

A Rule Based Lightweight Bengali Stemmer

Souvick Das
University of Calcutta
Kolkata, India
souvik.cs@hotmail.com

Rajat Pandit
West Bengal State University
Kolkata, India
rajatpandit123@gmail.com

Sudip Kumar Naskar
Jadavpur University,
Kolkata, India
sudip.naskar@gmail.com

Abstract

In the field of Natural Language Processing (NLP) the process of stemming plays a significant role. Stemmer transforms an inflected word to its root form. Stemmer significantly increases the efficiency of Information Retrieval (IR) systems. It is a very basic yet fundamental text pre-processing task widely used in many NLP tasks. Several important works on stemming have been carried out by researchers in English and other major languages. In this paper, we study and review existing works on stemming in Bengali and other Indian languages. Finally we propose a rule based approach that explores Bengali morphology and leverages *WordNet* to achieve better accuracy. Our algorithm produced stemming accuracy of 98.86% for Nouns and 99.75% for Verbs.

1 Introduction

Information retrieval is a very essential process to extract relevant data or documents systematically from big data collections. Inverted index is a crucial data structure used in almost all modern IR systems. All the words in the entire data collection is stemmed first before the inverted index is built. Thus stemming plays a very important role in IR (Kowalski, 2007).

Stemming is the process of reducing inflectional or derived variant forms of one word to its root form. Two major components of IR task is basically indexing and retrieval. Stemming not only enhances the recall rate of IR task but also reduces the index size significantly. Thus, it increases the efficiency of the information retrieval system.

Depending on the linguistic rules of a particular language, words of any natural language can be inflected in many ways. Bengali is one of the most morphologically decorated languages. Generally inflected words are generated from the root word by adding some suffixes. It is observed that one root word in Bengali may have more than 20 morphological variants. Another challenge is in finding root word from a compound word i.e a word can be formed by conjunction of more than one root words. A large number of notable works have been done on stemming techniques for different languages from the last couple of decades. Most of the approaches were first applied in English language and subsequently adapted in other languages. Different stemming approaches involve different techniques such as longest suffix matching, dictionary based look-up, co-occurrence computation etc. Several works (Faili and Ravanbakhsh, 2010; Urmi et al., 2016; Makhija, 2016) have been done based on these techniques in different languages. However, a very few papers utilize these techniques in Bengali language but failed to achieve over 95% of accuracy.

In this paper, we propose a *rule based technique* that utilizes rich volume of Bengali Grammar rules also involves *Bengali WordNet* (Dash et al., 2017) to achieve better accuracy. It is worth mentioning that we also verify the extracted stem word from our *rule based algorithm* with the help of modified *WordNet*. The sole purpose of the modification of the *WordNet* here is to reduce the complexity of the approach by circumventing the unnecessary prefix suffix removal of the word. The verification process and *WordNet* modification thus help us to achieve next level of higher accuracy. The following example helps to understand the proper usage of the *WordNet* in this approach. Suppose a word সরাইখানা (*Inn*) is

Proceedings of the 17th International Conference on Natural Language Processing, pages 400–408
Patna, India, December 18 - 21, 2020.

the root word and within this word one of the suffixes খানা is present. If we remove the suffix from the word, it will mean something else. Similar example can be seen for the word দুর্বার (*Strong*) where one of the suffix set র is present within the word. In Bengali stemming, filtering of root word and inflected word was a immense challenge. We overcome this problem by introducing the *WordNet* in the methodology. In the subsequent section, we present several cases where the modification of *WordNet* plays an important role.

The rest of the paper is structured as follows: Section 2 states the related works in this literature and brief review on them. Formation of different types of inflections in Bengali languages are analyzed in section 3. Section 4 elaborates different techniques involved in the proposed methodology and the entire methodology. Different resources that are used in the evaluation of the algorithm is described in section 5. Experiment and dataset used in the experiment is detailed out in section 6. Evaluation and results are presented in section 7. Finally we conclude this paper in section 8.

2 Related Work

There are two types of approaches in stemming mechanism. The first approach is called stemming in which affixes and suffixes are removed in order to extract the root word. The second approach is called *lemmatization*. Lemmatization requires a sound knowledge of the researcher in the particular language. As an example the word "Good" in English language has its variants "Better" and "Best". In order to get the *lemma* (root word) "Good" we need lemmatization. Lemmatization involves dictionary look up to solve this kind of unusual case of extracting root words.

Stemming technique for English language is well studied and several techniques have been developed. The very first stemming technique Lovins stemmer was proposed by Julie Beth Lovins in 1968 (Frakes and Fox, 2003). Lovins stemmer removes a suffix just in two major steps. It maintains 294 endings, 29 conditions and 35 transformation rules. Later, well known rule based Porters stemming (Porter, 1980) technique was proposed in 1980 which is basically a rule based algorithm. In its 8 steps it handles different cases of morphological and inflexional endings in different steps. An extension of the Lovins algorithm is known as Dawson stemmer (Jivani et al., 2011) was proposed which covers wide range of list of 1200 suffixes.

These above mentioned algorithms work very well for English language but we are more interested in different stemming algorithms for Indian languages. Ramanathan and Rao (2003) proposed a lightweight stemmer for Hindi in 2004 which removes the suffixes based on the longest suffix matching from a list of suffixes. They also developed a suffix list in Hindi language to enhance the performance of the stemming. Akram et al. (2009) proposed an affix-exception list based stemmer for Urdu language. They omit prefix and suffix from the word based on looking up the exception list of prefixes or suffixes. This stemmer finds the stem word based on lexical look up method. A successful look up ignores the stripping off of the prefix and suffix of a word. Hussain et al. proposed a stemming mechanism for Urdu language based on n-gram stripping model (Durrani and Hussain, 2010). Kumar and Rana (2011) developed a brute force algorithm to strip off suffixes in order to find stem words in Punjabi language. They have overcome the problem of over-stemming and under-stemming. The suffix stripping is replaced sometimes by suffix substitution. Islam et al. (2007) proposed a lightweight stemmer which strips off suffixes and finds the stem word for Bengali. The fundamental idea of this algorithm is to remove suffixes based on the longest suffix matching. They also maintain a list of possible suffixes for Bengali language. Paik et al. (2011) reported a simple corpus based unsupervised stemmer for Bengali Language. The algorithm uses some statistics collected from corpus analysis based on the co-occurrences between word variants. They generate a graph where nodes are the variants of word and an edge between them represents a co-occurrence. Das and Mitra (2011) used the Porter stemming technique in Bengali language. Majumder et al. (2007) developed a stemming technique based on statistical clustering based approach to discover equivalence classes of root words using some set of distance

Table 1: Comparison of different stemming techniques in Indian languages

Year	Language	Method	Author(s)	Description	Accuracy
2004	Hindi	A lightweight stemmer	Ramanathan et al.	Strip off the words endings from a suffix list on a 'longest match' basis.	88%
2009	Urdu	Affix-exception list based stemmer	Qurat-Ul-Ain Akram et al.	Stems the Urdu words using lexical lookup method (Assasband).	91.20%
2012	Urdu	Unsupervised approach to develop stemmer	Shahid Husain et. al.	n-gram stripping model	95.8%
2010	Punjabi	Brute Force Technique	Dinesh Kumar, Prince Rana	Suffix Stripping	80.73%
2007	Bengali	Yet Another Suffix Stripper	Majumder et al.	Statistical clustering based approach to discover equivalence classes of root words using some set of distance measures.	91.56%
2009	Bengali	A lightweight stemmer	Islam et. al.	Suffix Stripping	90.80%
2011	Bengali	A Fast Corpus-Based Stemmer	Jiaul H. Paik and Swapan Kumar Parui	A purely corpus based technique finds the equivalence classes of variant words in an unsupervised manner.	95%
2011	Bengali	Porter stemming technique	Suprabhat Das and Pabitra Mitra	Suffix Stripping	96.27%
2014	Bengali	Rule Based Bengali Stemmer	Redowan Mahmud. MD et. al.	Rule based suffix removal technique without using any list of suffixes.	88%
2016	Bengali	Bengali Lemmatizer(BenLem)	A. Chakrabarty and U. Garain	Reverse transformation based lemmatizer from surface words.	81.85%
2016	Bengali	A Neural Lemmatizer for Bengali	A. Chakrabarty et. al.	Neural network based lemmatizer using word2vec and CBOW	69.57%

measures. Mahmud et al. (2014) developed a rule based Bengali stemmer in 2014. This paper identify the occurrences of different inflections and their pattern. They developed rules to remove suffix from a inflected word without using any list of suffix list. Chakrabarty and Garain (2016) have designed a Bengali Lemmatizer (BenLem) which is able to handle both inflection and derivational morphology in Bengali language. In this approach, they used the FIRE Bengali News Corpus. It achieved 81.85% of accuracy in terms of resolving the inflected and derived words. Chakrabarty et al. (2016) proposed a neural network based lemmatizer which achieved 69.57% of accuracy. Thangarasu and Manavalan (2013) presented a literature review on stemming techniques for the Indian languages. Patel and Shah (2016) presented a literature review on unsupervised stemming techniques.

Table 1 shows different stemming techniques for Indian languages. In this comparison Das and Mitra (Das and Mitra, 2011) shows the highest accuracy of 96.27% for Bengali language with respect to other Indian languages and the lowest accuracy is 80.73% for Punjabi language proposed by Kumar and Rana (Kumar and Rana, 2011).

3 Inflections of words in Bengali language

Inflection of word is a process in which a word takes different forms. It may be based on tense, number and person. Different forms of the actual word are called inflected words. Bengali is one of the most morphologically decorated languages due to its wide range of inflected words. In this language inflections are mainly observed for verbs and nouns. Adjectives in Bengali can take only two suffixes - তর and তম, marking comparative and superlative adjectives, respectively, while adverbs in

Table 2: Possible Suffixes for Bengali Language

Tense	**1st Person & 2nd Person**	**2nd Person (Formally, Informally)**	**2nd Person (Informally for Junior Persons)**	**3rd Person (Formally)**	**3rd Person (Informally)**
Past Indefinite	লাম	লে, লেন	লি	লেন	লা, লো
Past Continuous	ছিলাম	ছিলে	ছিলি	ছিলেন	ছিল
Past Perfect	এছিলাম	এছিলে, এয়েছিলে, ইয়েছিলে এয়েছিলেন, এছিলেন, ইয়েছিলেন	ইয়েছিলি, এছিলি, এয়েছিলি	এছিলেন, এয়েছিলেন, ইয়েছিলেন	ইয়েছিল, এয়েছিল, এছিল
Present Indefinite	ই	এন	ইস	এন	এ
Present Continuous	চ্ছি, ছ	চ্ছ,ছ , চ্ছেন, এছেন	চ্ছিস, এছিস	চ্ছেন, এছেন	চ্ছে, এছে
Present Perfect	এছি	এছ, এছেন	এছিস	এছেন	এছে
Present Perfect Continuous	Not Applicable	এন	Not Applicable	উন	উক
Future Indefinite	ব (বো)	বে, বেন	বি	বেন	বে
Future Continuous	তেথাকব	তেথাকবেন	তেথাকবি	তেথাকবেন	তেথাকবে
Future Perfect	এথাকল	থাকবে	এথাকবি	এথাকবেন	এথাকবে
Future Perfect Continuous	Not Applicable	বেনওএন	তিস	বেন	বে
Habituatal Past	তাম	তে, তেন	তিস	তেন	ত

Bengali do not get suffixed. In this section we address the inflections of verbs and nouns. We also analyze the rules of inflections.

3.1 Inflections for Verbs

In Bengali language a verb is formed from the root-verb by joining some suffixes. As an example the root-verb of verb বলেছেন (*Told*) is বল and the suffix is এছেন. The inflections in verbs are varied according to tenses and the persons. The deviation of verb form according to the tense can be observed easily. For example the word বলছে (*Telling*), বলেছেন (*Told*) is deviation of actual root word বলা (*Tell*). Inflections are also noticed in case of informal and formal communications. For example the verb (Go) in the sentence "you (addressing younger one) are going" is presented as যাচ্ছিস and the verb (Go) in the sentence "you (addressing elder/ respected one) are going" is presented as যাচ্ছেন. Both of these words are infected form of যাওয়া (*Go*) and the root-verb is যা. One important point is essential to notice that the deviation of word যাওয়া (*Go*) is গিয়েছিলেন *(Went)* where there is no such linguistic interpretation. In such cases we maintain a mapping between root-verb and its possible deviations. We have listed out a number of possible suffixes in table 2 that are used to deviate a verb from its stem form. We apply rule based stemming mechanism to extract root-verb by omitting suffixes from the inflected word. A detailed procedure is illustrated in section 4. It is worth mentioning that the length of the root-verb in Bengali language is maximum 3. Root-verbs of different length is also presented in the table 3

3.2 Inflections for Noun

Noun inflections in Bengali language are limited. In case of verbal inflections the stem words can be changed sometimes but in case of Nominal inflection the stem word cannot be changed. Noun inflections are occurred to mention singular and plural forms of an object. Limited number of suffixes (Bhattacharya et al., 2005) such as `টি', `টা' ,`রা', `খানা', `খানি', `গুলো', `গুলি', `এরা', `রাজি', `রাশি', `পুঞ্জ', `সমূহ' etc. are added to the stem words. These suffixes are also added according to the representation of human being or other living things or non-living things. A number of noun inflections are presented in table 4. Example of different Noun inflections can be ছেলেটি (*The boy*), বইগুলি (*Books*), বৃক্ষাদি (*Trees*), পর্বতমালা (*Mountain range*) etc.

4 Proposed Methodology

We have discussed about various verb and noun inflections in Bengali Language till now. We also mentioned possible suffixes for deviation of a word and the different root-verb of different lengths of verbs. In this section, we illustrate our methodology to find stem of a verb.

It is easily observable that the number of rules for formation of different length of root-

Table 3: Different Types of Root-Verbs available in Bengali

Length	Category	Root-Verbs
1	Single Letter	হ, ক্ষ, ল, ঘ
1	Letter + আ	খা, ধা, পা ,যা, গা
1	Letter + ই	দি, নি
1	Letter +ু	শু, ধু, নু, etc
2	2 Letters	কর, কম, গড়, চল, পড়, জম etc around 100 root-verbs
2	letter + হ	কহ, সহ, বহ etc
2	্ is addded at last	কাট্, গাঁথ্, চাল্, আঁক্, কাঁদ্, বাঁধ্, লিখ্, কিন্, জিত্, ঘির্, ফির্, ভির্, চিন্, উঠ্, শুন্, ফুট্, খুঁজ্, খুল্, উড়্ etc almost 200 root-verbs
2	হ্ is added at last	গাহ্, বাহ্, নাহ্, চাহ্ etc
2	া is added at last	চড়া, কাটা, লাফা, চরা, ছড়া, ছরা, আগা, চালা, নাহা, গাহা, ফিরা, ছিটা, শিখা, ঝিমা, পিটা, মিটা, লুকা, ঘুরা, কুড়া, উঁচা, পুঁড়া, ধুয়া, ধোয়া, শোয়া, খোয়া, খোঁচা, গোছা, পৌঁছা, দৌড়া etc around 250 root-verbs
3	া is added at last	চটকা, সমঝা, কচলা, ধমকা, ছিটকা, হিঁচড়া, সিটকা, বিগড়া, দুমড়া, মুচড়া, উলটা, উপচা, ছোবলা, কোঁচকা, কোঁকড়া etc almost 150 root-verbs

Table 4: Different types of Noun Inflections

Objects	Singular	Plural
Human Beings	টা, টাকে, টি, টার, র, ের, কে etc	গুলো, গুলি, গুলোর, গুলির, রা, দের, দেরকে etc
Other Living or Non-Living Things	টা, টাকে, টি, টার, র, টাতে, টিতে, ের, ে etc	গুলো, গুলি, গুলোর, গুলির, এরা, জন, গুলিতে, গুলোতে, রাজি, রাশি, পুঞ্জ, সমূহ, বর্ণ, বৃন্দ, বর্গ, মালা, দি etc

verbs are limited. We can generate root-verbs of different length from an inflected word by applying the mentioned rules. For example from the inflected word খেয়েছেন (*ate*) we can generate possible root-verbs of different length upto 3. If we consider length 3 we get খয়ছা, খুয়ছা, খিয়ছা etc about 6 words. Similarly for length 2 we can generate words like খয়, খহ,খায়, খিয়, খুয় etc about 8 words and finally for length 1 we have খ, খা, খি, খু . Out of these all possible root-verbs only one will be matched with valid one (খা) and corresponding verb(খাওয়া) (*eat*) will be retrieved.

At this point we would like to illustrate our proposed methodology for stemming. The algorithm is presented in Algorithm 1. This algorithm takes one word possibly an inflected word $\mathbb{I}_{\mathbb{W}}$ and access Bengali WordNet. Before doing any kind of suffix removal it checks the word in the WordNet to confirm that the word is inflected or not. In some cases suffixes are present in a word and creates a new word. Removing this suffix from the word changes the intended meaning of the word. For example আধার (*Dark*), if we remove 'র' from the word, it will be আধা (*Half or Half pieces*). On the other hand if we consider word গাধার (Donkey's) and if we remove `র' then it will be গাধা (*Donkey*) which is valid suffix removal. So before doing suffix removal we should search that particular word in the WordNet. In the next stage, algorithm perform the stemming mechanism according to its category (Noun or Verb). In this phase root form of the verb is generated. The function $\mathbb{G}()$ generates the root-verb according to the lengths ranging from 3 to 1. It generates all possible root-verbs based on the first 3 letters of the inflected word by applying rules mentioned in table 3. If it does not find any matching valid root-verb, it continues to generate all possible root-verbs of length 2 and so on. As the number of rules are constant and very few, it does not take too much amount of time. If a match found then the corresponding verb of the root-verb is returned. In the previous step we have already filtered the root words having some suffixes within those root words. In this stage we can emphasis that the words will have intentional suffixes merged with its

Algorithm 1 Stemming Algorithm

Input $\mathbb{I}_{\mathbb{W}}$ is the word that is the inflected form of a verb or noun along with its parts of speech tagged. We maintain a database for list of nouns and verbs. $\mathbb{N}_{\mathbb{D}}$ is the database containing nouns and $\mathbb{V}_{\mathbb{D}}$ is the database containing verbs. We maintain a list of root-verbs, root form of Nouns are in $\mathbb{R}_{\mathbb{ND}}$ database and $\mathbb{R}_{\mathbb{VD}}$ accordingly. $\mathbb{S}_{\mathbb{F}}$ is the set of suffixes used to inflect a Noun.

OutputAfter applying the following procedure actual root word will be assigned to $\mathbb{R}_{\mathbb{W}}$.

procedure STEMMER($\mathbb{I}_{\mathbb{W}}$)
  $\mathbb{W}_{\mathbb{POS}} \leftarrow \mathbb{P}(\mathbb{I}_{\mathbb{W}})$
  if $\mathbb{W}_{\mathbb{POS}} = NOUN$ **then**
    if $\mathbb{S}(\mathbb{I}_{\mathbb{W}}, \mathbb{N}_{\mathbb{D}}) = true$ **then**
      $\mathbb{R}_{\mathbb{W}} \leftarrow \mathbb{I}_{\mathbb{W}}$
      $return(\mathbb{R}_{\mathbb{W}})$
    else
      $i \leftarrow 0$
      while $i > 4$ **do**
        $\mathbb{RW}_{\mathbb{P}} \leftarrow \mathbb{R}(\mathbb{I}_{\mathbb{W}}, \mathbb{S}_{\mathbb{F}}[i])$
        if $\mathbb{S}(\mathbb{RW}_{\mathbb{P}}, \mathbb{R}_{\mathbb{ND}}) = true$ **then**
          $\mathbb{R}_{\mathbb{W}} \leftarrow \mathbb{RW}_{\mathbb{P}}$
          $return(\mathbb{R}_{\mathbb{W}})$
        end if
        $i \leftarrow i + 1$
      end while
    end if
  end if
  if $\mathbb{W}_{\mathbb{POS}} = VERB$ **then**
    if $\mathbb{S}(\mathbb{I}_{\mathbb{W}}, \mathbb{V}_{\mathbb{D}}) = true$ **then**
      $\mathbb{R}_{\mathbb{W}} \leftarrow \mathbb{I}_{\mathbb{W}}$
      $return(\mathbb{R}_{\mathbb{W}})$
    else
      $length \leftarrow 3$
      while $length > 0$ **do**
        $\mathbb{RV}_{\mathfrak{d}} \leftarrow \mathbb{G}(length, \mathbb{I}_{\mathbb{W}})$
        for each $\mathcal{R}_{\mathcal{G}}$ in $\mathbb{RV}_{\mathfrak{d}}$ **do**
          if $\mathbb{S}(\mathcal{R}_{\mathcal{G}}, \mathbb{R}_{\mathbb{VD}}) = true$ **then**
            $\mathbb{R}_{\mathbb{W}} \leftarrow \mathbb{G}_{\mathbb{RV}}(\mathbb{RV}_{\mathfrak{d}}, \mathbb{R}_{\mathbb{VD}})$
            $return(\mathbb{R}_{\mathbb{W}})$
          end if
        end for
        $length \leftarrow length - 1$
      end while
    end if
  end if
end procedure

root. So we generate all possible root-verbs of length 3 then of length 2 and finally length of 1. It is worth mentioning that according to the Bengali grammar, the length of the root-verbs never exceed its length by 3. The next stage is set up for the Nouns.

We define an array of possible suffixes for Noun words mentioned in table 5 . Based on utilization of suffixes, we define a suffix stripping rules. Let us consider an inflected word ছেলেগুলোদেরকে (*to the boys*). In this word multiple suffixes are applied. So in general we first search for suffixes like কে, তে and remove those suffixes if present in the inflected word. Here in this example the inflected word ছেলেগুলোদেরকে becomes ছেলেগুলোদের. Now after that we search ◌ে, র ,এরা, য়, রা and remove those suffixes if available. Now the word becomes ছেলেগুলোদে. Again we search for দে, কা, টা, টি and remove the one is present in the word. So the word is now ছেলেগুলো (*Boys*). After that the algorithm searches for জন, গুলো, গুলি, খানা and removes the appropriate one. In this stage we get ছেলে (*Boy*) which is the root word. Furthermore, the algorithm will search for রাজি, রাশি, বর্ণ, পুঞ্জ, বৃন্দ, বর্গ, সমূহ but at any stage if the word can be found in the WordNet we return that word as the root form of the inflected word.

Another major notable limitation of Bengali WordNet is that, WordNet does not contain deviated words a lot. It is certainly much difficult to identify all linguistic deviation of a word and store them all in WordNet. A word can be spoken or written in different way and that is deviated from one region to another region of a country. As an example, the deviated form of the word বলা (*Telling*) can be বলতে or বলতা. Inhabitants of various regions of West Bengal (A state of India) use the word 'বলতে' and on the other hand some inhabitants of some regions of West Bengal use the word 'বলতা'.

At this point we briefly illustrate the steps involved in the proposed algorithm. The algorithm takes inflected word ($\mathbb{I}_{\mathbb{W}}$) as input. In the next step function $\mathbb{P}()$ extracts the parts of speech ($\mathbb{W}_{\mathbb{POS}}$) of $\mathbb{I}_{\mathbb{W}}$. If it is a "NOUN" then a function $\mathbb{S}()$ searches the Noun database ($\mathbb{N}_{\mathbb{D}}$) for a match. If a match found, it means the inflected word has a meaning itself hence there is no need of stemming. Otherwise we can strip off different suffixes depending upon the presence of suffixes within the inflected word. We iterate through the suffix set $\mathbb{S}_{\mathbb{F}}$ and a function $\mathbb{R}()$ searches different possible suffixes within the inflected word and removes them. $\mathbb{R}()$ returns a word $\mathbb{RW}_{\mathbb{P}}$ in every iteration that does not contain i^{th} suffix set of $\mathbb{S}_{\mathbb{F}}$. $\mathbb{RW}_{\mathbb{P}}$ then searched in the "Root Noun" database $\mathbb{R}_{\mathbb{ND}}$ using function $\mathbb{S}()$. Whenever a match found the word is returned as the root form of the inflected word.

If $\mathbb{W}_{\mathbb{POS}}$ is "VERB" then again the search function $\mathbb{S}()$ searches the word in verb database ($\mathbb{V}_{\mathbb{D}}$) for a match. A match indicates

Table 5: Suffix Array

Index	1	2	3	4	5
Suffixes	কে, ত	◌, র ,এরা, য়, রা	দে, কা, টা, টি	জন, গুলো, গুলি, খানা	রাজি, রাশি, বর্ণ, পুঞ্জ, বৃদ, সমূহ

that the word has its own meaning. So we return the word as it was. Otherwise a function $\mathbb{G}()$ will generate all possible root-verb from the inflected word and store them in set $\mathbb{RV}_{\eth}$. In first iteration it takes first 3 characters of the inflected word and make all possible root-verbs using the predefined rules. Then it takes 2 characters and so on. As in Bengali grammar root-verbs can have of length maximum of three, we start finding root-verbs in descending order. Every element within the set $\mathbb{RV}_{\eth}$ is checked in the root-verb database $\mathbb{R}_{\mathbb{VD}}$. At any point if a match found then the corresponding stem verb ($\mathbb{R}_{\mathbb{W}}$) is returned.

5 Resources Used

Bengali WordNet: Bengali WordNet is a part of IndoWordNet[1]. Bengali WordNet is a lexical database for Bengali words and it contains around 61 thousand Bengali words along with the Synsets. We imported the Bengali WordNet in MySQL database. We created a separate table for all the root-verbs corresponding to the Bengali verbs.

TDIL Corpus: For the corpus, we used the Technology Development for Indian Languages (TDIL) corpus (Jha, 2010) in this work.

6 Experiment

6.1 Dataset

We tested our algorithm on a testset of randomly chosen 500 sentences from the TDIL Bengali corpus of health and tourism domain articles. In this test dataset there are 2,756 unique content words containing 1304 Nouns, 1230 Verbs, 54 Adverbs and 168 Adjectives. Since in Indian languages, nouns and verbs get highly inflected, we concentrated on the stemming of nouns and verbs in Bengali. We used the *Stanford Bengali POS Tagger* to assign POS tag to each word of the testset. Finally we manually verified the POS tags and corrected the wrongly assigned tags.

[1]http://tdil-dc.in/indowordnet/

Table 6: System performance

Category	#Words	#Correctly Stemmed	Accuracy
Noun	1274	1234	96.86
Verb	1230	1227	99.75

6.2 Implementation

We implemented our algorithm in Python 3.6 and MySQL. In the very first step each sentence of the corpus is scanned by our python script. The Natural Language Tool Kit (nltk) package has been used to accomplish the necessary preprocessing tasks. Within the nltk package we have used Stanford Bengali POS Tagger to tag parts of speech for each word of the sentence. Then the sentence is tokenized into set of words. Another python script has been used to remove stop words listed by TDIL. At this point we use our actual python program to implement our proposed algorithm. It takes the tokenized words along with tagged parts of speech. We have used MySQL to store the Bengali WordNet. Whenever a searching in the WordNet is required, a particular module is responsible to search a particular word in the WordNet.

7 Result and Analysis

Our algorithm works for Verbs and Nouns. Out of 1230 verbs it successfully stems 1217 Verbs. In the other hand it successfully stems 1274 Nouns. Our algorithm fails for 43 words due to lack of words in Bengali WordNet. The accuracy of our algorithm shows 96.86% for the Verbs and 99.75% for Nouns. The accuracy of our algorithm is shown in table 6. The comparison of accuracy of different stemming techniques are presented in Figure-1. In figure-1 it is shown that our technique gives better accuracy than state-of-art stemming techniques in Bengali language.

In our approach we validate inflected words before applying our proposed rule based suffix removal technique. This validation technique

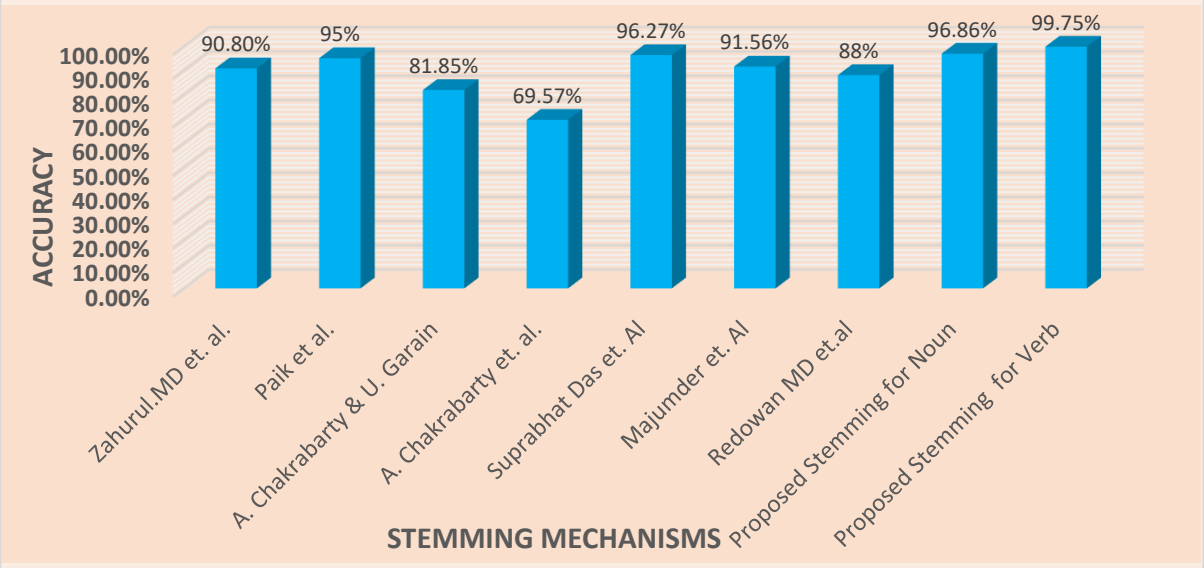


Figure1
Comparison of Accuracy of Different Stemming Techniques

is done by searching the word in the WordNet to check whether the word is inflected or not. We incorporate the rules of Bengali grammar to understand how verbs are formed from the root-verbs. Subsequently we observed that there are very limited rules for inflections of nouns. We extract the root-verbs from the inflected verbs by finding the combinations of suffixes and root-verbs of different lengths. Root words are also extracted from inflected nouns by applying step by step suffix removal.

The mechanism extracts actual root word from the inflected word and verifies it with the WordNet entry. There are some advantages and disadvantages in this approach. One of the major advantages is that the extracted root word will always be correct. This validation enhances the correctness of the extraction of root words.

One vital limitation of this entire mechanism is that, our algorithm rely on WordNet. There may be a situation where our algorithm extracts correct root word but it is not present in the WordNet and the extracted root word will be discarded.

8 Conclusions and Future Work

We have proposed a rule based algorithm for stemming verbs and nouns in Bengali. Incorporation of WordNet adds an extra degree in validation and extracting root words from inflected words. Bengali grammar rules have been used to find root-verbs of verbs efficiently. We have covered almost all kinds of root-verbs and possible suffixes to create a root word from an inflected verb.

References

Qurat-ul-Ain Akram, Asma Naseer, and Sarmad Hussain. 2009. Assas-band, an affix-exception-list based urdu stemmer. In *Proceedings of the 7th workshop on Asian language resources*, pages 40–46. Association for Computational Linguistics.

Samit Bhattacharya, Monojit Choudhury, Sudeshna Sarkar, and Anupam Basu. 2005. Inflectional morphology synthesis for bengali noun, pronoun and verb systems. In *Proc. of the National Conference on Computer Processing of Bangla (NCCPB 05)*, pages 34–43.

Abhisek Chakrabarty, Akshay Chaturvedi, and Utpal Garain. 2016. A neural lemmatizer for bengali. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2558–2561.

Abhisek Chakrabarty and Utpal Garain. 2016. Benlem (a bengali lemmatizer) and its role in wsd. volume 15, pages 1–18. ACM New York, NY, USA.

Suprabhat Das and Pabitra Mitra. 2011. A rule-based approach of stemming for inflectional and derivational words in bengali. In *Students' Technology Symposium (TechSym), 2011 IEEE*, pages 134–136. IEEE.

Niladri Sekhar Dash, Pushpak Bhattacharyya, and Jyoti D Pawar. 2017. The wordnet in indian languages. Springer.

Nadir Durrani and Sarmad Hussain. 2010. Urdu word segmentation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 528–536. Association for Computational Linguistics.

Heshaam Faili and Hadi Ravanbakhsh. 2010. Affix-augmented stem-based language model for persian. In *Proceedings of the 6th International Conference on Natural Language Processing and Knowledge Engineering (NLPKE-2010)*, pages 1–4. IEEE.

William B Frakes and Christopher J Fox. 2003. Strength and similarity of affix removal stemming algorithms. In *ACM SIGIR Forum*, volume 37, pages 26–30. ACM.

Md Islam, Md Uddin, Mumit Khan, et al. 2007. A light weight stemmer for bengali and its use in spelling checker. BRAC University.

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

Anjali Ganesh Jivani et al. 2011. A comparative study of stemming algorithms. volume 2, pages 1930–1938.

Gerald J Kowalski. 2007. *Information retrieval systems: theory and implementation*, volume 1. Springer.

Dinesh Kumar and Prince Rana. 2011. Stemming of punjabi words by using brute force technique. volume 3, pages 1351–1357.

Md Redowan Mahmud, Mahbuba Afrin, Md Abdur Razzaque, Ellis Miller, and Joel Iwashige. 2014. A rule based bengali stemmer. In *Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on*, pages 2750–2756. IEEE.

Prasenjit Majumder, Mandar Mitra, Swapan K Parui, Gobinda Kole, Pabitra Mitra, and Kalyankumar Datta. 2007. Yass: Yet another suffix stripper. volume 25, page 18. ACM.

Sangita D Makhija. 2016. A study of different stemmer for sindhi language based on devanagari script. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 2326–2329. IEEE.

Jiaul H Paik, Dipasree Pal, and Swapan K Parui. 2011. A novel corpus-based stemming algorithm using co-occurrence statistics. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 863–872. ACM.

Miral Patel and Apurva Shah. 2016. An unsupervised stemming: A review. volume 14, page 476. LJS Publishing.

Martin F Porter. 1980. An algorithm for suffix stripping. volume 14, pages 130–137. MCB UP Ltd.

Ananthakrishnan Ramanathan and Durgesh D Rao. 2003. A lightweight stemmer for hindi. In *the Proceedings of EACL*.

M Thangarasu and R Manavalan. 2013. A literature review: stemming algorithms for indian languages.

Tapashee Tabassum Urmi, Jasmine Jahan Jammy, and Sabir Ismail. 2016. A corpus based unsupervised bangla word stemming using n-gram language model. In *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*, pages 824–828. IEEE.

End-to-End Automatic Speech Recognition for Gujarati

Deepang Raval, Vyom Pathak, Muktan Patel, Brijesh Bhatt
Computer Engineering Department, Dharmsinh Desai University, Nadiad
deepangraval2012@gmail.com, angerstick3@gmail.com,
muktan123@gmail.com, brij.ce@ddu.ac.in

Abstract

We present a novel approach for improving the performance of an End-to-End speech recognition system for the Gujarati language. We follow a deep learning based approach which includes Convolutional Neural Network (CNN), Bi-directional Long Short Term Memory (BiLSTM) layers, Dense layers, and Connectionist Temporal Classification (CTC) as a loss function. In order to improve the performance of the system with the limited size of the dataset, we present a combined language model (WLM and CLM) based prefix decoding technique and Bidirectional Encoder Representations from Transformers (BERT) based post-processing technique. To gain key insights from our Automatic Speech Recognition (ASR) system, we proposed different analysis methods. These insights help to understand our ASR system based on a particular language (Gujarati) as well as can govern ASR systems' to improve the performance for low resource languages. We have trained the model on the Microsoft Speech Corpus, and we observe a 5.11% decrease in Word Error Rate (WER) with respect to base-model WER.

1 Introduction

ASR is the process of deriving the transcription (word sequence) of an utterance, given the speech waveform. Speech Recognition has been an active area of research for many decades. Initial work in ASR was based on statistical modeling techniques like Hidden Markov Model (HMM) (Baker, 1975) and used phonemes to represent distinct sounds that make up the word. With the rise of Deep Learning based techniques and the increasing availability of data, the End-to-End speech recognition systems started showing competitive results. Initial deep learning based ASR models, based on Recurrent Neural Network (RNN) and CTC (Graves et al., 2006), overcame the issues of statistical systems and provided the mapping of variable length input to output. With the further advancements in algorithms and resources, various complex deep learning architectures have been introduced for an effective End-to-End speech recognition system. End-to-End speech recognition for low resource languages has not gained significantly from the advancements in deep learning due to lack of training data compared to other high resource languages. Linguistic diversities[1] also makes it difficult to adopt models across languages.

In this paper, we present a speech recognition system for the Gujarati Language. Gujarati is a rich language consisting of 34 consonants and 13 vowels. While the more number of vowels may reduce the homophones, more number of consonants may increase the ambiguity.

The key contributions of this paper are as follows,

- We have adopted the state of the art ASR model described in (Amodei et al., 2015) for the Gujarati Language.
- We present a novel approach of combining two language models, 4-gram word-level language model (WLM) and bi-gram character-level language model (CLM) to improve performance of prefix decoding.
- We propose a *Spell Corrector BERT* based post-processing technique to correct erroneous prediction and further improve the performance of the ASR System.

[1] http://www.cs.cmu.edu/ ytsvetko/jsalt-part1.pdf

Proceedings of the 17th International Conference on Natural Language Processing, pages 409–419
Patna, India, December 18 - 21, 2020.

The proposed system reduced the WER to 65.54% from the initial 70.65%. We analyzed the system using the testing hypothesis and derived many useful insights on the performance of the system as well as the cause of the errors in the hypothesis. We analyzed that the errors produced in the Gujarati language are mainly because of interchanging/mismatching diacritics (‘િ’, ‘ી’, etc.), consonants (‘ધ’, ‘જ’, ‘ચ’, etc.), independents (‘આ’, ‘અ’, ‘ઈ’, ‘ઈ’, etc.) and some homophones.

The remaining of the paper is organized as follows, Section 2 describes the Literature Survey of ASR system architectures. The proposed approach is described in Section 3. Section 4 constitutes the experiments conducted and its observations are followed in Section 5. Section 6 provides the conclusion of our work.

2 Related Work

Since the first ASR circuit developed by Bell Laboratories (Davis et al., 1952) in the 1950s, ASR has remained an active area of research. In early 1960's (Kenichi et al., 1966) presented a phoneme based speech recognition which involved the first use of speech segmenter in different portions of the input utterances. (Vintsyuk, 1968; Sakoe and Chiba, 1978) introduced the concept of the non-uniform time scale for alignment of speech patterns (dynamic wrapping). Both of these works lead to the Viterbi Algorithm (Viterbi, 1967) which had been an indispensable technique in ASR for decades. By the mid-1970s, the basic ideas of applying fundamental pattern recognition technology to speech recognition, based on Linear Predictive Coding (LPC) (Atal and Hanauer, 1971) methods, were proposed by Itakura (Itakura, 1975). CMU's Harpy System (Lowerre and Reddy, 1976), was the first ever system to use the Finite State Network (FSN) to reduce computation for matching in Speech Recognition. However, methods which optimized the resulting FSN did not come about until the early 1990's (Mohri, 1997), which were limited to small to medium vocabulary electronic based solutions for ASR.

The earlier approaches of Electronics based ASR were eventually replaced by statistical approaches with the introduction of HMM based speech recognition. The basic implementation of HMM based speech recognition model was first published in 1975 by Baker (Baker, 1975) at CMU. Further work on HMM continued with the introduction of first ever use of HMM for continuous speech recognition in 1976 (Jelinek, 1976). As the research continued, the HMM model was tried with various machine learning techniques including the HMM/ANN architecture in 1990 (Bourlard and Wellekens, 1990), HMM/GMM architecture in 1997 (Rodríguez et al., 1997) and HMM/SVM architecture in 1998 (Golowich and Sun, 1998). This wave for HMM continued till the introduction of RNN based approaches in early 2005.

The above approaches had major drawbacks such as

- It requires high task-specific knowledge, e.g. to design the state models for HMMs.
- It requires fairly complex parameter tuning as the pipeline contains multiple configurations.

(Graves et al., 2006) introduced a novel method for training RNNs to label unsegmented sequences directly, using CTC, thereby eliminating the above drawbacks and creating an End-to-End ASR system. With further enhancements in algorithms and resources, deep learning based End-to-End ASR systems got better and better and they started outperforming traditional ASR systems (Graves and Jaitly, 2014; Hannun et al., 2014). End-to-End ASR systems with encoder-decoder have shown competitive results (Chan et al., 2016). The RNN encoder-decoder paradigm uses an encoder RNN to map the input to a fixed-length vector and a decoder network to expand the fixed-length vector into a sequence of output predictions (Cho et al., 2014; Sutskever et al., 2014). Adding an attention mechanism to the decoder greatly improves the performance of the system, particularly with long inputs or outputs.

As we saw End-to-End ASR gives great results but at the cost of large data required to train it, which is not feasible for low resource languages. According to *Interspeech 2018, Low Resource Automatic Speech Recognition Challenge*[2], TDNN-based systems (Ped-

[2]https://tiny.cc/Interspeech2018

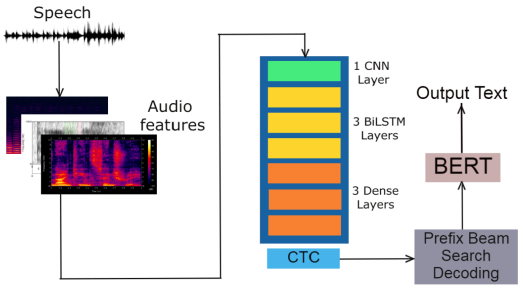


Figure 1: End-to-End Automatic Speech Recognition Process

dinti et al., 2015) are efficient in modelling long temporal context and performed well even in the low-resource setting (Fathima et al., 2018; Pulugundla et al., 2018). Even with the smaller amount of data, with some enhancements, End-to-End systems showed promising results (Billa, 2018).

The work similar to our approach are presented in *speech recognition primer*[3], and *Gujarati Automatic Speech Recognition*[4]. The first approach is based on a combination of CNN (Chua and Yang, 1988) and BiLSTM (Schuster and Paliwal, 1997) the latter approach uses a combination of 3 Gated Recurrent Units (GRUs). The first approach is designed for English, while the second is for Gujarati.

Our approach differs from the above two as follows,

- The model architecture described in our paper constitutes 1 CNN - 3 BiLSTM - 3 Dense layers.
- We present a more effective approach to decode the output using the prefix beam search algorithm along with the combination of the language model.
- Our approach introduces a post-processing technique to improve the performance of the system even more.

BERT is a neural network-based technique for natural language processing pre-training. The pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications (Devlin et al., 2018). Multilingual-BERT uses a representation that is able to incorporate information from multiple languages (Pires et al., 2019).

[3]https://github.com/apoorvnandan/speech-recognition-primer

[4]https://github.com/niteya-shah/Gujarati-Automatic-Speech-Recognition

3 Proposed Approach

Figure 1 describes the End-to-End speech recognition system proposed in the paper. This section describes the processing involved in the various stages.

3.1 Audio feature

We have used mel-frequency cepstral coefficients (MFCC) (Motlıcek, 2002) as features to represent the input audio signal. We convert the input audio signal into MFCC. The dimension of these features are *(Time_Steps, MFCCs)* and for the given batch size it is of dimension *(Batch_Size, Time_Steps, MFCCs)*. These features serve as the input to the deep learning model.

3.2 Model Architecture

The model architecture incorporates four major components: CNN layer, BiLSTM layer, Dense Layer, and CTC. Each component has its own importance and components like CNN layer, BiLSTM layer, and Dense layer have to be tuned as per the size of the input data.

Convolutions in frequency and time domains, when applied to the spectral input features prior to any other processing, can slightly improve ASR performance (Abdel-Hamid et al., 2012; Sainath et al., 2013). It also attempts to model spectral variance due to speaker variability, which is another reason to use the first layer as convolution (Amodei et al., 2015). We have used a single 1D-convolution layer with 200 filters with *ReLU* activation function with kernel size 11 and stride value as 2. Features extracted are then passed to a deep BiLSTM RNN (Schuster and Paliwal, 1997).

We have used 3 BiLSTM layers, each layer consisting of 200 BiLSTM units (400 LSTM units) and *tanh* as the activation function of each unit. When provided input from the con-

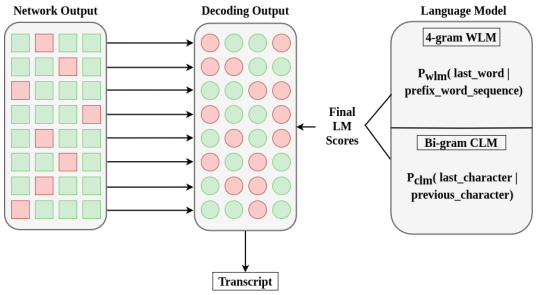


Figure 2: Working of decoding algorithm using WLM and CLM

volution layer, the BiLSTM layer gives us output as *(Batch_Size, Convolved_Time_Steps, LSTM_blocks)*. Features extracted are then passed to a DNN (Hinton et al., 2006), which consists of 3 layers where the first 2 layers consist of 200 units. The number of units in the last layer is equal to the number of characters in the languages.

While training, a common technique for mapping variable-length audio input to variable-length output is the CTC algorithm (Graves et al., 2006) coupled with an RNN. The CTC-RNN model performs well in End-to-End speech recognition with grapheme outputs (Graves and Jaitly, 2014; Hannun et al., 2014; Maas et al., 2014, 2015). Given the network outputs, CTC maximizes the probabilities of the correct labelings. The CTC objective function is differentiable thus the network can then be trained with standard back-propagation through time (Werbos, 1990).

3.3 Decoding

We propose an enhanced language model based prefix decoding which uses our custom built 4-gram word-level language model (WLM) and a bi-gram character-level language model (CLM) (Brown et al., 1992). Here we refer to it as *Prefix with LMs'*. Both of these models were created using the whole *Gujarati Wikipedia*[5]. Using prefix decoding with two language models (WLM and CLM) makes decent corrections, as it introduces language constraints at both word and character scope. We have compared this approach with greedy decoding (Maas et al., 2014) and prefix decoding (Maas et al., 2014).

As shown in the figure 2, the output from the network is passed through the prefix decoding algorithm which incorporates a WLM and a CLM. The WLM will score the last word given in the prefix word sequence. This score's influence can be controlled by WLM weight (wlm). Similarly, CLM will score the last character to be appended given its previous character. This score's influence on the new prefix is controlled by CLM weight (clm). We encompassed the insertion bonus by multiplying the count of words in prefixes to avoid bias towards shorter prefixes. To control the influence of insertion bonus we used beta (β).

3.4 Post Processing

We propose a BERT based post-processing technique to improve the output of speech recognition systems. The BERT model is used to correct the spelling of the predicted output words. Here we term this technique as *Spell Corrector BERT*. Figure 3 describes the working of sentence correction algorithm using BERT. For a sentence, we iterate through all the words during which we find the replacements for the current word by finding its corresponding zero, one, or two edit substitutes from the Wikipedia corpus. Using the list produced by this approach, we can verify that if the word predicted is correct, it would be already present in the list and needs not to be replaced.

If the current word is not present in the replacements list, then that word is replaced with $[MASK]$ and the sentences are generated by replacing the $[MASK]$ with the word replacements from the replacements list. Further, the sentences are tokenized and passed to the BERT model. As an output, BERT returns the list of replaced words and their respective probabilities w.r.t. the sentence. From this list, we select K word replacements with the highest probability and append this list of word replacements to the *output_list*. Once all the words are iterated, a final *output_list* is generated which contains a list of all words with at most K replacements for each word. Given a sentence containing 3 words, અમદાવાદ એપોર્ટ પર where અમદાવાદ and એપોર્ટ are incorrect, this process gives the *output_list*= [[અમદાવાદ, અમદાવાદમા,...] , [એરપોર્ટ, પોર્ટ,...] , [પર]]. This *output_list* is passed to a combinator which generates sen-

[5]https://gu.wikipedia.org/wiki/

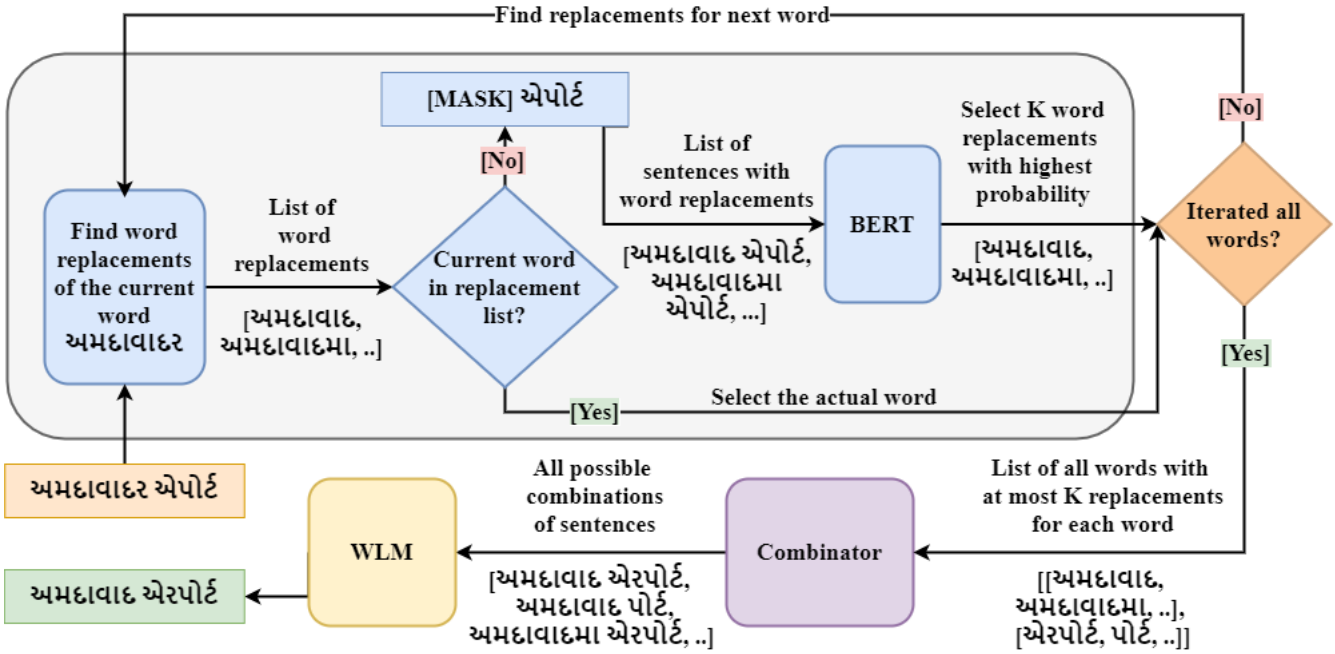


Figure 3: Working of Spell Corrector BERT

tences by combining the words. If there is a list contain a single word in it, it is selected as it is and if there is a list of word replacements then various combinations are produced using the combinator. The output of this process will be *output_sentences*= [અમદાવાદ એરપોર્ટ પર , અમદાવાદ પોર્ટ પર , અમદાવાદમા એરપોર્ટ પર ,...]. To select the best sentence out of all the sentences, a WLM is used for sentence scoring and the sentence with the highest score is selected as the $final\_output$ = અમદાવાદ એરપોર્ટ પર.

4 Experiments

4.1 Dataset

We have used Microsoft Speech Corpus available for Gujarati[6] which contains approximately 22,807 training examples and 3,075 testing examples. The dataset contains an eclectic collection of speakers where the length of a single audio utterance is 6.35±2.33 seconds. Out of 22,807 training examples, we have used 16,000 utterances (≈28.2 Hours) for training and 4,807 (≈8.5 Hours) utterances for validation and all the testing examples i.e. 3,075 (≈5.4 Hours) utterances for inferencing.

[6]https://msropendata.com/datasets/7230b4b1-912d-400e-be58-f84e0512985e

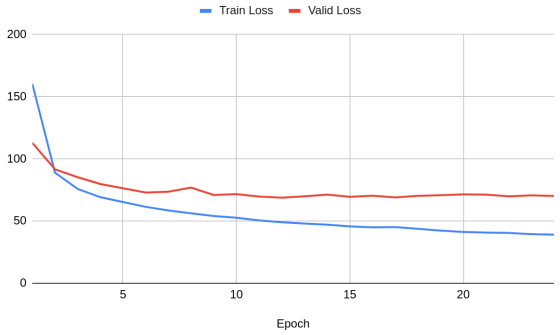


Figure 4: Training and Validation Loss for the model

4.2 Training

The training data, as well as validation data, was divided into 7 batches and the model was trained for 24 epochs and 92 hours on T4 GPU with 16 GB of GPU-memory. The model consists of 2,744,676 total parameters. We used *Adam* optimizer for gradient descent, and for calculating the loss we used CTC loss function.

4.3 Decoding

We have used Gujarati data scraped from Wikipedia containing 2,501,841 words with a vocabulary size of 163,170 words. We used this corpus to create the statistical word-level language model and produced 1,570,614 4-grams. We also used the same corpus to create the sta-

Table 1: Sentence and its corresponding output through various decoding techniques.

Actual	અમદાવાદ એરપોર્ટ પર સુરક્ષાને લઈ તમામ તૈયારીઓ કરી દેવાઈ છે
Greedy	અમદાવાદાર પર પણ સુરક્ષાઅને લઈને તમમ ટેરે કરી જવોાય છે
Prefix with LMs'	અમદાવાદાર પર પણ સુરક્ષાને લઈને તમમ કેરે કરી જવાય છે
Prefix with LMs' & Spell Corrector BERT	અમદાવાદ પર પણ સુરક્ષાને લઈને તમામ કેરે કરી જવાય છે

Table 2: Distribution of single letter error words

Technique	Consonants	Diacritic	Independents
Greedy	66.41% (1,962)	28.23% (834)	5.34% (158)
Prefix with LMs'	66.52% (1,824)	27.97% (767)	5.5% (151)
Prefix with LMs' & Spell Corrector BERT	52.90% (829)	38.67% (606)	8.4% (132)

Table 3: Techniques and their corresponding WER

Techniques	Word Error Rate (%)
Greedy	70.65
Prefix without Language Model	69.95
Prefix with WLM	69.53
Prefix with CLM	68.64
Prefix with LMs'	68.23
Prefix with LMs' & Spell Corrector BERT	65.54

tistical character-level language model to create bi-grams for each alphabet of the Gujarati language. For prefix decoding, the beam width was taken as 50 and all other parameters were decided using cross-validation. The algorithm of *Prefix with LMs'* recorded a 2.42% decrease in WER w.r.t. system using greedy decoding technique.

4.4 Post Processing

Spell Corrector BERT is used to further improve the output produced by *Prefix with LMs'*. We have used a pre-trained BERT multilingual model by Google[7] combined with a 4-gram WLM as the core components of *Spell Corrector BERT*. The algorithm of *Spell Corrector BERT* recorded a 2.69% decrease in WER w.r.t. standalone *Prefix with LMs'*. The table 3 shows the comparison of WER for different techniques.

[7]https://github.com/google-research/bert/blob/master/multilingual.md

5 Observation

5.1 Comparison of various decoding and post-processing techniques

We have tested the performance of the decoding technique and post-processing by observing the distribution and frequency of the single letter error words. Table 1 shows a sample testing sentence as well as the hypothesis generated by our model using various decoding techniques and post-processing techniques. Table 2 describes the distribution of single-letter error words observed in different approaches. Here, the count of errors due to consonants/diacritics/independents w.r.t. the total count of single-letter error words in each decoding technique is shown as a percentage. It shows that, *Prefix with LMs'* and *Prefix with LMs' & Spell Corrector BERT* post-processing, both help in reducing single letter error words.

Table 2 also shows count of single letter error words. Subsequently, from this count we can conclude that, percentage decrease of the error in consonant, diacritic and independents using *Prefix with LMs'* is 7%, 8% and 4% respectively w.r.t greedy decoding, while using *Prefix with LMs' & Spell Corrector BERT*, the percentage decrease of the error in consonant, diacritic and independents is 57.74%, 27.00% and 16.45% respectively w.r.t. greedy decoding.

We observe that, with a lesser number of

incorrect characters, *Spell Corrector BERT* either retains WER or in the majority of the cases, will improve WER significantly. Table 4 shows sample sentences with a different number of erroneous words for comparison of the performance of *Spell Corrector BERT*.

5.2 System Analysis

The system analysis is performed on the model hypothesis decoded using *Prefix with LMs' & Spell Corrector BERT*. We have evaluated the performance of the proposed model on 3,075 test examples. Out of total erroneous words, 7.19% words have one letter error with similar sounding alphabets of letters interchange. e.g. 'શ' → 'સ', 'ઇ' → 'ઈ', 'િ' → 'ી'. The interchange of consonants/diacritics/independents is due to the factors like, noise, channel variability, speaker variability, anatomy of the vocal tract, speed of speech, regional and social dialects, homophones (Forsberg, 2003). Any incorrect word in the sentence is replaced on the basis of probability and hence WLM, CLM, or BERT is not solely responsible for the selection of any word, it is the combination of the probabilities that results in the final output.

5.2.1 Error due to consonants/independents/diacritics

Table 5 displays examples of words which have a single-letter error due to consonants, independents, and diacritics. *Ref.* depicts the actual word in the sentence, *Hyp.* denotes the output word, *Ref. Freq.* shows the count of reference word in the corpus, *Hyp. Freq.* is the count of hypothesis word in the corpus, *Ref.* → *Hyp.* shows the character that is replaced and *Type* shows the type of error in the inference word. From this table, we can observe that, despite the hypothesis word being infrequent in the corpus, the similar sounding letters in the reference word gets replaced. This advocates the idea that the replacement of the similar sounding letters from the words is also one of the factors inducing the errors in the system, irrespective of the words' frequency in the corpus.

Out of the total 1,567 one letter error words, 52.90% errors are due to single consonant mismatch. The top three incorrectly predicted consonants are 'ક', 'ર', 'ત', with frequencies 102, 92, and 79 respectively. Together they contribute to 32.93% of total errors due to consonants.

In figure 5, the connection between two consonants represents the error of interchanging/misplacing these consonants with each other. For example, the connection between 'શ' and 'સ' indicates that these consonants are generally interchanged/misplaced with each other, in a word predicted by the system.

Out of the total 1,567 one letter error words, 8.40% errors are due to single-independent mismatch. The top three incorrectly predicted independents are 'ઈ', 'ઇ', 'અ', with frequencies 81, 21, and 13 respectively. Together they contribute to 87% of total errors due to independents. We observe that (['ઇ', 'ઈ'],['ઉ', 'ઊ']) are more vulnerable to being misplaced/interchanged.

Out of the total 1,567 one letter error words, 38.67% errors are due to single-diacritic mismatch. The top three incorrectly predicted independents are 'ી', 'ે', 'ા', with frequencies 213, 120, and 76 respectively. We observe that frequency of diacritic 'ી' and 'ે'is greater than the sum of the frequencies of remaining diacritics, and they constitute to 67% of total incorrectly predicted diacritics.

The output ([શરૂ, સરૂ],[સાડા, સાળા],[કોઈ, કોએ]) are interesting cases as the hypothesis words are not present in the corpus. The predicted output misplaced (['શ', 'ડ', 'ઈ']) with similar sounding (['સ', 'ળ', 'અ']) respectively without any prior knowledge of the word. This is due to the character by character prediction approach of our model.

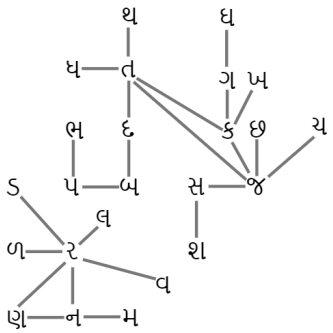


Figure 5: Interchanging consonants which results in erroneous prediction.

Table 4: Sample sentences for comparing performance of BERT

Actual	અમદાવાદ એરપોર્ટ પર સુરક્ષાને લઈ તમામ તૈયારીઓ કરી દેવાઈ છે
At most one error per word	અમદાવદ એપોર્ટ પર સુરક્ષામે લઈ તમાન તૈયરીઓ જરી દેરાઈ છે
Spell Corrector BERT Output	અમદાવાદ એરપોર્ટ પર સુરક્ષાને લઈ તમામ તૈયારીઓ કરી દેવાઈ છે
At most two errors per word	અમદવા એરપર્ પર સરક્ષને લઈ તમ તૈયરઓ કરી દેવ છે
Spell Corrector BERT Output	અમદાવાદ એરપોર્ટ પર સુરક્ષાને લઈ તમે તૈયારીઓ કરી દેવ છે
At least 1 word with error greater than 2	અમદાવાદાર પર પણ સુરક્ષાઅને લઈને તમમ ટેરે કરી જવોાય છે
Spell Corrector BERT Output	અમદાવાદ પર પણ સુરક્ષાને લઈને તમામ કેરે કરી જવાય છે

Table 5: Examples of words which have single letter error

Ref.	Hyp.	Ref. Freq.	Hyp. Freq.	Ref.→Hyp.	Type
શરૂ	સરૂ	282	0	શ → સ	Consonant
ત્યારે	ક્યારે	377	15	ત → ક	Consonant
ધરાઈ	ધરાઇ	9	15	ઈ → ઇ	Independent
ઊભેર્યું	ઉભેર્યું	2	11	ઊ → ઉ	Independent
પ્રારંભિક	પ્રારંભીક	4	0	િ → ી	Diacritic
ચૂંટણીમાં	ચુંટણીમાં	50	12	ૂ→ ુ	Diacritic

5.2.2 Error due to homophones

Out of a total of 606 words that had a single diacritic error, only 2.97% of errors were due to homophones which is a small fraction of the total amount of errors in the inference from the ASR system. This might be because Gujarati is mostly a phonetic language with only a few exceptions. Also, the number of alphabets (vowels and consonants) in Gujarati are more than that in English. By reducing diacritic errors, we can resolve errors due to homophones. This helps us to understand that our system is not much affected by error due to homophones. Table 6 shows the incorrectly predicted homophones.

5.2.3 Effect of word frequency on error

To understand the effect of frequency on error, we calculated the frequency of the predicted words in the test dataset. We categorized the words into three different categories based on the correctness of the word referenced to their occurrence in the testing dataset. The three different categories are,

- ACPW: Words that are always predicted correctly.
- AIPW: Words which are always predicted incorrectly
- CAIPW: Words which are predicted correctly as well as incorrectly at times.

Count of ACPW, AIPW and CAIPW is 1,069, 7,809 and 1,604 respectively. Mean frequencies/occurrences of ACPW, AIPW and CAIPW is 1.11, 1.34, and 15.38 respectively. Words which are ACP, AIP and CAIP in testing, are shown in training with a mean frequency/occurrence of 4.75, 4.86, and 77.81 respectively with a count of 857, 5,764 and 1,594 respectively. This gives us a rationalization for the fact that our system is able to learn from the utterances shown in the training and can infer unseen examples too.

Table 6: Examples of words which are incorrectly predicted homophones

Reference Word	કર્તા (Actor)	રવિ (Sun)	પીતા (Drinking)
Hypothesis Word	કરતા (Than)	રવી (Winter Crop)	પિતા (Father)

Table 7: Sample words which are predicted correctly as well as incorrectly

Words in Testing	Total Count	Wrong count	Right count	Correctness(%)
આજે	81	32	49	60.49
રાજકોટના	2	1	1	50.00
તાકાત	2	1	1	50.00
વિભાગ	7	4	3	42.86
આવે	67	24	43	64.18
Average				53.50

We also analyzed the correctness of the words from the set CAIPW. 8.32 out of 15.38 mean frequencies of CAIPW are correct and the remaining 7.06 out of 15.38 are incorrect. Table 7 shows some examples of the correctly as well incorrectly predicted words with the amount of correctness. This gives an explanation for how words are predicted correctly as well as incorrectly with the same proportion.

5.2.4 Error due to half-conjugates

This type of error occurs due to the mismatch in the speed of utterance. The fast-conjugate error occurs when a word is uttered too quickly but the hypothesis word is predicted slow, e.g., (મુખર્જી → મુખરજી). When a word is uttered slowly but the hypothesis word is predicted fast then this type of error is called slow-conjugate error (ગયો → ગ્યો). Out of total erroneous predicted words, 2.4% of them have half-conjugate error and out of those 2.4% words, 10% words consist of pure half-conjugate erroneous words. This justifies the fact that the error due to half-conjugate is trivial and thus the variation in speaker speed is not a significant factor due to which error occurs in inference by our system.

6 Conclusion

In this paper, we have presented an End-to-End speech recognition system for Gujarati. We propose a prefix decoding technique that uses two language models to improve the performance of the system. We have also used a BERT based spelling corrector model in a post-processing step to further improve the performance. We observe that the proposed approach reduces the overall WER by 5.11%.

While deep learning models require a lot of training data for better results, in this paper we showed that without increasing the training data we can improve the performance. This is particularly important for a resource-constrained language like Gujarati. We are optimistic that with an increase in data our optimizations would perform even better.

We explored and analyzed the inferences from our ASR system to gain key insights which consist of checking the correctness of the word error due to consonants, diacritics, independents, half-conjugates, and homophones. These insights can help to understand our ASR system based on a particular language (Gujarati) as well as can govern ASR systems' to improve the performance for low resource languages.

Acknowledgement

Param Shavak supercomputer was used to perform some experiments of this research work. We are thankful to GUJCOST and Department of Science and Technology, GoG for establishing supercomputer facility.

References

O. Abdel-Hamid, A. Mohamed, H. Jiang, and G. Penn. 2012. Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4277–4280.

Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse H. Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Y. Hannun, Billy Jun, Patrick LeGresley, Libby Lin, Sharan Narang, Andrew Y. Ng, Sherjil Ozair, Ryan Prenger, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Yi Wang, Zhiqian Wang, Chong Wang, Bo Xiao, Dani Yogatama, Jun Zhan, and Zhenyao Zhu. 2015. Deep speech 2: End-to-end speech recognition in english and mandarin. *CoRR*, abs/1512.02595.

Bishnu S Atal and Suzanne L Hanauer. 1971. Speech analysis and synthesis by linear prediction of the speech wave. *The journal of the acoustical society of America*, 50(2B):637–655.

J. Baker. 1975. The dragon system–an overview. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):24–29.

Jayadev Billa. 2018. Isi asr system for the low resource speech recognition challenge for indian languages. pages 3207–3211.

H. Bourlard and C. J. Wellekens. 1990. Links between markov models and multilayer perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(12):1167–1178.

Peter F Brown, Vincent J Della Pietra, Peter V Desouza, Jennifer C Lai, and Robert L Mercer. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–480.

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964. IEEE.

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 translation. *arXiv preprint arXiv:1406.1078*.

L. O. Chua and L. Yang. 1988. Cellular neural networks: theory. *IEEE Transactions on Circuits and Systems*, 35(10):1257–1272.

Ken H Davis, R Biddulph, and Stephen Balashek. 1952. Automatic recognition of spoken digits. *The Journal of the Acoustical Society of America*, 24(6):637–642.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Noor Fathima, Tanvina Patel, C Mahima, and Anuroop Iyengar. 2018. Tdnn-based multilingual speech recognition system for low resource indian languages. In *INTERSPEECH*, pages 3197–3201.

Markus Forsberg. 2003. Why is speech recognition difficult.

Steven E. Golowich and Don X. Sun. 1998. A support vector/hidden markov model approach to phoneme recognition, in. In *Center for Media Technology (RCMT), School of Creative Media, City University of Hong Kong, Hong Kong*, pages 125–130.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural 'networks. volume 2006, pages 369–376.

Alex Graves and Navdeep Jaitly. 2014. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page II–1764–II–1772. JMLR.org.

Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. 2014. Deep speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567.

Geoffrey E. Hinton, Simon Osindero, and Y. Teh. 2006. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554.

Fumitada Itakura. 1975. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on acoustics, speech, and signal processing*, 23(1):67–72.

F. Jelinek. 1976. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–556.

Maeda Kenichi, Sakai Toshiyuki, and Doshita Shuji. 1966. Phonetic typewriter system. US Patent 3,265,814.

B Lowerre and R Reddy. 1976. The harpy speech recognition system: performance with large vocabularies. *The Journal of the Acoustical Society of America*, 60(S1):S10–S11.

Andrew Maas, Ziang Xie, Dan Jurafsky, and Andrew Ng. 2015. Lexicon-free conversational speech recognition with neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 345–354, Denver, Colorado. Association for Computational Linguistics.

Andrew L. Maas, Awni Y. Hannun, Daniel Jurafsky, and Andrew Y. Ng. 2014. First-pass large vocabulary continuous speech recognition using bi-directional recurrent dnns. *CoRR*, abs/1408.2873.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational linguistics*, 23(2):269–311.

Petr Motlıcek. 2002. Feature extraction in speech coding and recognition. Technical report, Technical Report of PhD research internship in ASP Group, OGI-OHSU,< http ….

Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. 2015. A time delay neural network architecture for efficient modeling of long temporal contexts. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual bert? *CoRR*, abs/1906.01502.

Bhargav Pulugundla, Murali Karthick Baskar, Santosh Kesiraju, Ekaterina Egorova, Martin Karafiát, Lukás Burget, and Jan Cernockỳ. 2018. But system for low resource indian language asr. In *INTERSPEECH*, pages 3182–3186.

Elena Rodríguez, Belén Ruíz, Ángel García-Crespo, and Fernando García. 1997. Speech/speaker recognition using a hmm/gmm hybrid model. In *Audio- and Video-based Biometric Person Authentication*, pages 227–234, Berlin, Heidelberg. Springer Berlin Heidelberg.

Tara Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. 2013. Deep convolutional neural networks for lvcsr. pages 8614–8618.

Hiroaki Sakoe and Seibi Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49.

M. Schuster and K. K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Taras K Vintsyuk. 1968. Speech discrimination by dynamic programming. *Cybernetics*, 4(1):52–57.

Andrew Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269.

P. J. Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Deep Neural Model for Manipuri Multiword Named Entity Recognition with Unsupervised Cluster Feature

Jimmy Laishram
Department of CSE
Manipur Technical University
Imphal, Manipur, India
jimmy_l@mtu.ac.in

Kishorjit Nongmeikapam
Department of CSE
IIIT, Manipur
Imphal, Manipur, India
kishorjit@iiitmanipur.ac.in

Sudip Kumar Naskar
Department of CSE
Jadavpur University
Kolkata, West-Bengal, India
sudip.naskar@cse.jdvu.ac.in

Abstract

The recognition task of Multi-Word Named Entities (MNEs) in itself is a challenging task when the language is inflectional and agglutinative. Having breakthrough NLP researches with deep neural network and language modelling techniques, the applicability of such techniques/algorithms for Indian language like Manipuri remains unanswered. In this paper an attempt to recognize Manipuri MNE is performed using a Long Short Term Memory (LSTM) recurrent neural network model in conjunction with Part Of Speech (POS) embeddings. To further improve the classification accuracy, word cluster information using K-means clustering approach is added as a feature embedding. The cluster information is generated using a Skip-gram based words vector that contains the semantic and syntactic information of each word. The model so proposed does not use extensive language morphological features to elevate its accuracy. Finally the model's performance is compared with the other machine learning based Manipuri MNE models.

1 Introduction

Multi Word Named Entity (MNE) is a part of Multiword Expression (MWE) which is an ordered group of words that can exist independently and carries different meaning as opposed to its constituent word (Nongmeikapam and Bandyopadhyay, 2011). Accurate recognition of such MNE plays a vital role in various NLP tasks such as POS tagging (Nongmeikapam et al., 2011b), Chunking (Nongmeikapam et al., 2014), NER classification (Singh and Bandyopadhyay, 2010). Manipuri, being a highly inflectional language where affixes define the nature of the words (Choudhury et al., 2004), machine recognition of MNE presents a challenging task for NLP researchers. In Manipuri, the majority of researches on MNE classification are done using machine learning approaches such as CRF (Nongmeikapam and Bandyopadhyay, 2010, 2011; Nongmeikapam et al., 2011a) and SVM (Singh and Bandyopadhyay, 2010). These researches use extensive morphological features to obtain accurate recognition of the MNE. Such morphological features include affixes, context words, digit features etc. For an agglutinative and low-resource language, the inflectional nature amounts to the large Out of Vocabulary (OOV) words, thus making any sequence labelling task or morphological feature creation tasks difficult.

Word embedding is a distributed representation of text in low dimensional real valued vectors and are known to contain semantic or syntactic information and has shown to be an effective feature for many natural language classification tasks of English language (Wang et al., 2015). Word embedding has also been extremely useful for Chinese language processing (Yin et al., 2016), Japanese language processing (Kitagawa and Komachi, 2017) and for some Indian languages processing (Ajay et al., 2016; Bhattacharya et al., 2016). In Manipuri, the effectiveness of word embeddings to any NLP tasks, till this date remains unanswered. The general attempt to work in Manipuri NLP task using embedding follows the idea used in English i.e. to learn the embedding of a word from the context. Unlike English, Manipuri words are a composite of complicated structure with several affixes producing OOV words. Regardless of its context, affixes to a word plays a major role in defining the semantic meaning (Choudhury et al.,

Proceedings of the 17th International Conference on Natural Language Processing, pages 420–429
Patna, India, December 18 - 21, 2020.

2004). In this paper, a Manipuri MNE classification task is attempted using a bi-directional Long Short Term Memory (Bi-LSTM) with Skip-gram word embeddings. The usage of extensive morphological features for deep neural network is avoided, as these features would require vector representations which may lead to considerable large input layer to update. To segregate the Manipuri MNE words based on the semantics, a K-means algorithm based cluster information of each word is added as a feature to the LSTM model.

1.1 Manipuri Multi Word Named Entity

Manipuri is an Indian language which is highly agglutinative in nature, tonal, has reduplicated words and no gender marking. It belongs to the Tibeto-Burman languages spoken mostly in North-Eastern region of India which includes Manipur and Assam. The Manipuri MNEs can be decomposed into multiple lexemes and displays lexical, syntactic, semantic, pragmatic idiomatic behaviour. These encompasses all the multi-word named entities such as a person's name, location name etc (Nongmeikapam et al., 2014). Some examples of MNEs are given in table 1 below:

MNE Details	Example
Beginning of Person's Name	আরডি (RD)
Internal of Person's Name	মেহতা (Mehta)
Beginning of Location Name	ন্যু (New)
Internal of Location Name	দিল্লিগী (Delhi-gi)
Beginning of Organization Name	ঠিয়েটর (Theatre)
Internal of Organization name	সেন্টর (Center)

Table 1: Manipuri MNE

Nongmeikapam and Bandyopadhyay (2010) have identified various challenges of MNE recognition task as described below:

- Manipuri lacks capitalization of named entities unlike English or other European languages.
- The MNE inflections can be found in the language because of nominal suffixes and pronominal prefixes.
- Due to free word order in the language, the MNE can appear in subject or object position.
- Many Named Entities (NEs) can appear in a dictionary that carry different meanings which creates a homonym effect.
- Manipuri is a resource constraint language. Annotated corpus, name dictionaries, morphological analyser, POS tagger etc are not readily available.

1.2 Motivation

The agglutinative and inflectional nature of Manipuri has poised a challenge for any computational processing task. Manipuri is a language where NLP resources such as annotated corpus, accurate morphological analyzer etc are not readily available. Above all, the present deep neural network NLP algorithms and language modelling techniques have not been proven its efficiency for language such as Manipuri. The introduction of deep neural network in the Manipuri NLP task such as MNE classification can provide a beneficial step towards the challenges faced in POS tagging, NER etc.

2 Related Works

Notable amount of Manipuri MWE classification research have been done using traditional machine learning approach such as CRF (Conditional Random Field) and SVM (Support Vector Machine). Nongmeikapam and Bandyopadhyay (2010) reported an improvement of Manipuri MWE identification using CRF and Reduplicated Multi-Word Expressions (RMWE). The MWE identification model uses a dictionary and rule-based approach to first detect the complete, mimic, partial, double and echo RMWEs and then prepare a model training set using features such as affix information, POS information word frequency and word length. The CRF model attained a performance F-Score measure of 72.24%.

Nongmeikapam et al. (2011a) conducted a research for identification of RMWE using CRF with various features. The features were stem words, number of affixes, stemmed affixes, POS of surrounding words, surrounding words, length of the word, word frequency and digit features. With the features, the CRF classification model predicted with Recall, Precision and an F-Score measure of

92.91%, 91.90% and 92.40% respectively.

Nongmeikapam and Bandyopadhyay (2011) conducted a research on CRF based MWE identification with Genetic Algorithm (GA) based feature selection method. In genetic algorithm, the MWE features are represented as genes in a chromosome which are binary valued (1 or 0). Selection of a feature is done when the gene value is 1. Random crossover is performed to select the best possible combination of features. To avoid chromosome repetition, random mutation is performed. As a fitness function, three fold cross validation technique is used. Using the technique, the best features were surrounding words, affixes, surrounding POS, word length and word frequency. The CRF model with the aforementioned GA based selected features, attained an increase in F-score by 2.91% as compared to baseline CRF model without GA based feature selection. Overall the model performed with F-Score= 73.74%, Precision= 86.84% and Recall= 64.08%.

Singh and Bandyopadhyay (2010) proposed a web based Manipuri corpus for Multiword NER and RMWE identification model using SVM. For classification purpose, the best features were selected for SVM classifier. The features were context word, word affixes, MNE and RMWE information, digit features, infrequent words, word length and POS information. The model was trained over 1235 sentences with 28629 words and predicted with an F-Score measure of 93.96% for MNE and 94.07% for RMWE.

3 System Design

The Manipuri MNE classification is shown in Figure 1. Being a low resource language with limited information on morphological features, extraction of necessary information for a particular word is crucial as these information will help the model in accurate MNE classification. In this research, two models have been implemented for Manipuri MNE classification. They are

1. A baseline model with Bi-LSTM deep neural networks with Word embedding using Skip-gram with default POS embedding.

2. A model 1 in which the cluster information is added as additional feature to the baseline model.

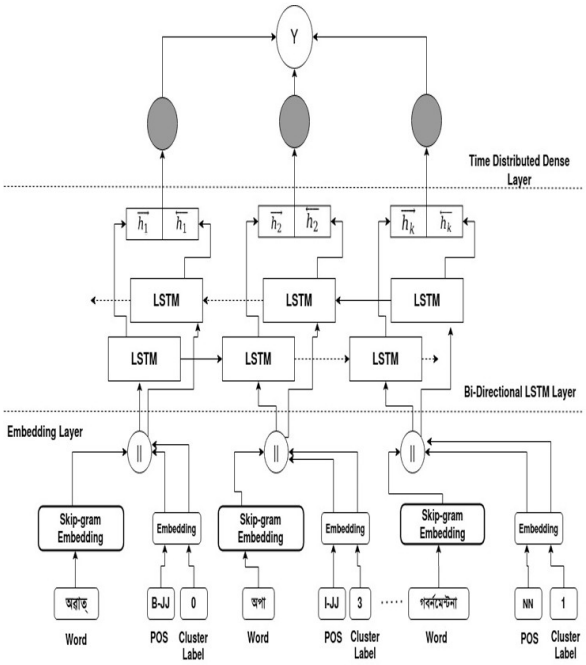


Figure 1: Manipuri MNE Classification System Architecture

The use of word embedding, feature embedding and Bi-LSTM are described in the following subsections.

3.1 Word Embeddings

As deep neural network function on real-valued vectors, it is essential that the input words ($S = x_1, x_2, x_3...x_t$ where x_t are the word sequence in a sentence S) are converted to a D-dimensional real-valued word vector that carries semantic and syntactic information. The Skip-gram (Mikolov et al., 2013a) word embedding algorithm is used to create the word vectors because for a small corpora with infrequent words, the skip-gram embedding can represent rare words (Mikolov et al., 2013b) with precision as compared to other word embedding techniques. The skip-gram model is described below.

Skip-gram Model: (Mikolov et al., 2013a) introduced Skip-gram model for vector representation of large amount of unstructured words without the need of dense matrix multiplication. The objective of the Skip-gram model is to find the word representation that can predict surrounding words in a sentence.

The Skip-gram model takes in a sequence of words $W = w_1, w_2, w_3....w_N$ and generates the context word $C = c^1, c^2...., c^k$ on the basis of the center word w_i.

Given a sequence of Manipuri words $w_1, w_2, w_3, ..., w_N$, the Skip-gram model maximizes the average log probability P as given below:

$$\frac{1}{N}\sum_{(n=1)}^{N}\sum_{(-m\leq j\leq m, m\neq 0)} \log(p(w_{n+m}|w_n)) \quad (1)$$

where m is the size of the training context which can be a function of the center word w_i. When $p(w_{n+m}|w_n)$ is put to a softmax function, we get:

$$p(w_c|w_n) = \frac{e^{v_c.v_w}}{\sum_{w=1}^{W} e^{(v_w.v_c')}} \quad (2)$$

where v^w and v^c are the input and output vectors of word vocabulary W and context words C respectively. Now putting the probability of equation 2 in equation 1, we get:

$$\sum_{(w_n\in W, w_c\in C)} log(p(w_c|w_n))$$
$$= \sum_{(w_n\in W, w_c\in C)} (log e^{v_c.v_w} - log \sum e^{v_c'.v_w}) \quad (3)$$

Using the above described Skip-gram model (actual implementation is described in 4.3), a word embedding is obtained that encodes the semantic and syntactic information to its real-values vectors. The hyper-parameter for the model is as follows: Minimum word count = 3, window size= 3 and embedding dimension= 120.

3.2 Word Cluster Formation

To elevate the accuracy, cluster information is added as an additional feature to train the dataset using the K-means algorithm and the Skip-gram word vectors as shown in Figure 2 where each word is assigned to a specified cluster using the Euclidean distance similarity.

The K-means clustering is performed on a normalized word vectors X_{norm}, as cosine similarity and euclidean similarity are connected linearly and bears same result in clustering

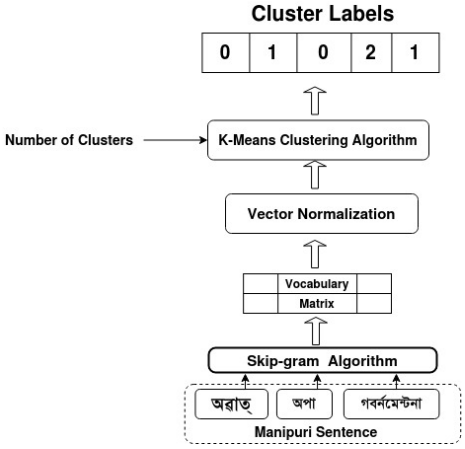


Figure 2: Word Clustering using K-Means Algorithm

(Qian et al., 2004). The normalized word vector is given by:

$$X_{Norm} = \frac{X}{max(X)} \quad (4)$$

With the normalized word vectors $X_{norm} = (x^1_{norm}, x^2_{norm}, ..., x^n_{norm})$, each word is assigned to its appropriate k-cluster category using the Euclidean distance similarity measure which is given by:

$$d_{Euclidean}(p, q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \quad (5)$$

Where p and q are data points. Finally the objective function with k random data points chosen to be the initial centroid is given by:

$$argmin(\sum_{i=1}^{k}\sum_{v\in V} d_{Euclidean}(v, s_i)) \quad (6)$$

3.3 POS and Cluster Embedding

As neural network work with tensors, each additional feature needs to be in a matrix form, called embedding matrix where it relates each index of the object with its translation to tensors. Selecting a vector of a specific object can be translated into a matrix product in the following way:

$$v_i = \begin{cases} 1, if i \neq Object_{index} \\ 0, if i = Object_{index} \end{cases} \quad (7)$$

$$[Object_{vector}]_{Dim\times 1} = [M]_{Dim\times words}.[\vec{V}]_{(pos/cluster)\times 1} \quad (8)$$

Where $\vec{V}$ is the one-hot vector that determines which word needs to be translated. And M is the embedding matrix. Two matrices will be created each for POS and cluster feature.

3.4 LSTM Layer

In this Manipuri MNE recognition research, the recognition task is done using a bi-directional LSTM RNN. The purpose of choosing the LSTM network is because of its capability to overcome the diminishing gradient problem when the input sequence is large. In our MNE classification research, it has been found that the longest sequence is of 120 Manipuri words as shown in Figure 3 which led us to choose the LSTM RNN.

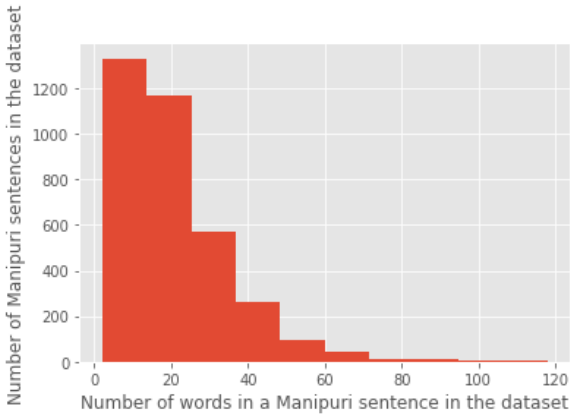


Figure 3: Manipuri Sentence length

LSTM is an Recurrent Neural Network (RNN) that works on a sequential data(Lample et al., 2016). A sequence of data) $\{w_1, w_2, ...w_n\}$ as input (Concatenation of Word, POS, Cluster and Affix vectors) and return another sequence $\{h_1, h_2, ...h_n\}$ that represents some information at every time step in the input which is given by:

$$h_t = lstm(h_{t-1}, [E(w_t)||e(w_t^{pos}, p_t)]) \quad (9)$$

$[E(w_t)||e(w_t^{pos}, p_t)] = x_t$ is the embedding where $E(w_t)$ is the word embedding for the word w_t using Skip-gram, $e(w_t^{pos})$ is the POS embedding and $e(p_t)$ cluster embedding for the Manipuri word w_t . The symbol || in equation 9 represents concatenation of embedding vectors. The LSTM architecture is shown in 4.

LSTM(Reddy et al., 2018) consists of three gates that control the proportion of the input

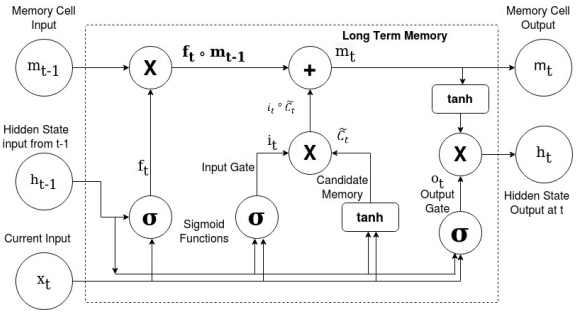


Figure 4: Long Short Term Memory Architecture

to give to the memory cells, and the proportion from the previous states to forget gate, that helps to overcome the diminishing gradient problem faced by RNN. At any given time t over Manipuri input sequence, three gates composite the LSTM unit cell:

1. An input gate i_t with the corresponding weight matrix: W_i and b_i, which is mathematically represented as:

$$i_t = \sigma(W_i[h_{t-1} + x_t] + b_i) \quad (10)$$

2. A forget gate f_t with corresponding weight matrix: W_f and b_f. Mathematically f_t is represented as:

$$f_t = \sigma(W_f[h_{t-1} + x_t] + b_f) \quad (11)$$

3. An output gate o_t with corresponding weight matrix: W_o and b_o. Mathematically o_t is represented as:

$$o_t = \sigma(W_o[h_{t-1} + x_t] + b_o) \quad (12)$$

where σ is the sigmoid function

All of these aforementioned gates are set to generate a certain state using the current input x_t, the state h_{t-1} from the previous step and current state of this cell, for the decisions whether to forget the memory stored, to take the input or to output the state generated. $\tilde{C}$ is the new candidate to be added to the new state which is given by:

$$\tilde{C}_t = \tanh(W_t[h_{t-1} + x_t] + b_c) \quad (13)$$

The current state c_t will be generated by calculating the weighted sum using both previous cell and current information generated by the

current cell. The following equation provides the current state c_t:

$$c_t = i_t \tilde{C}_t + f_t . c_{t-1} \tag{14}$$

Finally, the hidden state h_t to the next LSTM unit is calculated as:

$$h_t = o_t \tanh(c_t) \tag{15}$$

As Manipuri is a context dependent language, it is beneficial to have access to the future and past contextual information which led to implementation of bidirectional LSTM modifier which is given by:

$$\overrightarrow{h_t} = lstm(\overrightarrow{h_{t-1}}, x_t))$$
$$\overleftarrow{h_t} = lstm(\overleftarrow{h_{t-1}}, x_t))$$

3.5 Output layer

The output layer of the model consist of Time-Distributed Dense function with softmax activation. This function allows us to apply the same function across every output over the time. Finally the softmax classifier calculates a probability distribution over the sequence labels.

4 Experimental Setup

4.1 Dataset

The dataset for Manipuri MWE classification is collected from a leading newspaper agency "The Sangai Express[1]" in Manipur with 76526 words. The appropriate POS and MNE tags are manually annotated and the dataset is split into training and testing set in 80:20 ratio. Further, 1 Million unannotated Manipuri dataset is used for the Skip-gram model training. Table 2 describes the dataset used in training and testing of the proposed Manipuri MWE model:

Details	Values
Number of Sentences	3504
Total number of words	76526
Number of distinct words	16297
Maximum number of words in a sentence	120
Multi-word Named Entities	12421

Table 2: Manipuri Dataset details

[1]https://www.thesangaiexpress.com/

4.2 Hyper-parameter details

For Manipuri MNE classification model, different hyper-parameters settings were used for bi-directional LSTM training as described in Table 3. As deep neural network tends to overfit, spatial dropout and recurrent dropout are used as regularization in the model whose value is described in the table 3 below:

Hyper-parameters	Values
Bi-LSTM units	100
Bi-LSTM batch size	32(Bengio, 2012)
Spatial1D Dropout	30%
Recurrent Dropout	10%
Activation Function	Sigmoid
Loss Function	binary cross-entropy
Optimizer	RMSprop Learning Rate= 0.0001, $\rho = 0.9$

Table 3: Manipuri MNE Classification Hyper-parameters

Ruder (2016) suggests the use of the Gradient Descent optimizer:RMSprop, to overcome the radically diminishing learning rates during training. Graves (2013) published the first use of RMSprop optimizer in recurrent neural network and suggest a default value of learning rate = 0.0001 and decay constant $\rho = 0.9$.

4.3 The Skip-gram model

The Skip-gram model as shown in figure 5, takes in a pair of inputs words for each training example ([input word w_i, target word c_i]) having unique identifier which is then passed to an embedding layer initialized with random weights. It is then passed to a merge layer

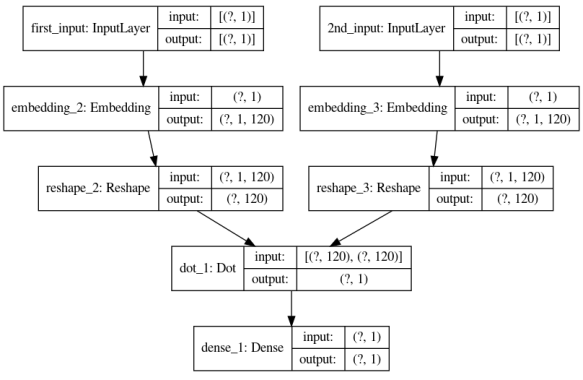


Figure 5: Skip-gram Model summary

to compute the dot product of these vectors where a sigmoid layer predicts the output

$$Y = \begin{cases} 1, if w_i \ and \ c_i \ is \ relevant \\ 0, if \ otherwise \end{cases} \tag{16}$$

The loss is leveraged by the mean squared error loss and performs back-propagation with each epoch to update the embedding layer.

5 Results and Evaluation

The main aim of this research is to focus on the addition of features and their effects on the final result of MNE classification as compared to the base models. We have used the Precision, Recall and F-Score evaluation metrics to measure the performance of the models.

5.1 Result of the Skip-gram model

Figure 6 shows the word similarity plot using Euclidean distance measure of the Skip-gram model. The circled areas contains some of

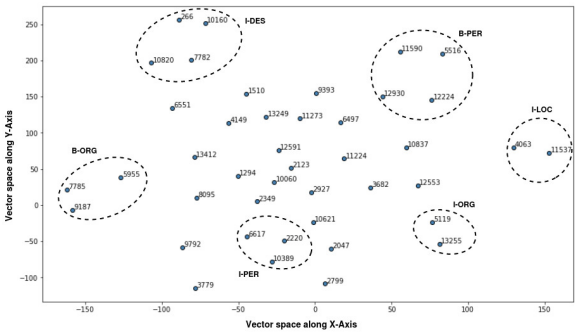


Figure 6: Euclidean Distance Similarity among words of Skip-Gram Model. The numbers in the figure represents index of the Manipuri MNE in the word vector

the MNE words identified by its index, that have similar euclidean distance, positioned near each other in the vector space.

Types of Relationship	Word	Similar word
	Word Pair	
Organization	কাউন্সিলগী Council-gi	পার্টিসু Party-su
Person Name	থৌনাউজম Thounaojam	লোকেন Loken
Location	টিডিম Tiddim	মোংবুং Moibung
Designation	মিনিষ্টরনা Minister-na	সেক্রেটরি Secretary

Table 4: Types of Word Relationship among word pairs

Table 4 shows the word-word similarity among the MNEs. The proposed Ski-gram model was found that the similarity of word representation go beyond the syntactic regularities with 89% of MNEs having found similarly distant in the vector space. The agglutinative nature of the language affects the Skip-gram model, as affixes define the inflection of the words (Bhat and Ningomba, 1997).

In the case of Time, Date and Currency MNEs, the Skip-gram model cannot find the similar word as these words are basically numbers and identifier words for Time পূং (poong), Currency লূপা (Lupa) and Date তাং (Tang) are unique and does not occur frequently in the dataset.

5.2 Optimal number of Clusters

The MNE classification models (as described in section 3) uses word cluster information generated from the Skip-gram model with K-means clustering algorithm. The clustering is to segregate the words into cluster and added as an additional feature to the LSTM model training. To select the optimal number of cluster for the k-means algorithm to perform clustering, the Silhouette analysis is performed for cluster number 3 to 10.

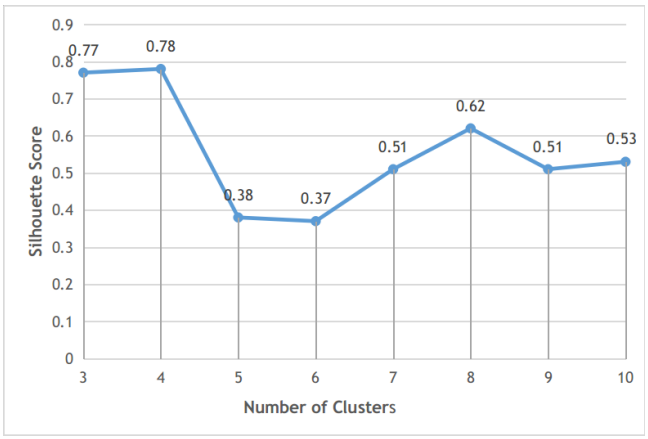


Figure 7: Silhouette Score for the clusters

From the Figure 7, it can be seen that the $s(4) = 0.78$ which is closer to 1, which led us to choose the number of clusters = 4.

5.3 MNE LSTM model Output

With the embedding weights generated from the Skip-gram model and the cluster information using the K-means algorithm, the MNE Bi-LSTM model is created. The result of the MNE classification using Bi-LSTM and Skip-gram embedding is shown in figure 8. The baseline model attained an average F-Score measure of 81.47% in classifying the MNE.

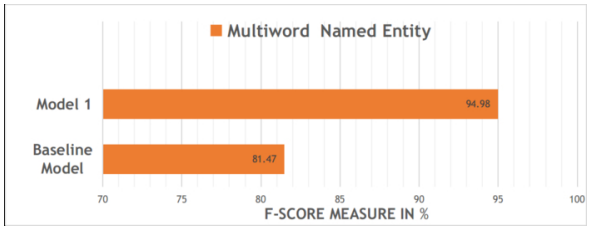


Figure 8: Average F-Score measure of Manipuri MNE models

With the addition of cluster information during the training, the MNE classification F-Score measures is increased to 94.98%. The fine grain classification report of the baseline model with cluster features is shown in Table 5.

MNE	P	R	F	Support
MW Person name	1.00	1.00	1.00	128
MW Location name	0.9408	0.9310	0.9358	52
MW Date/Time	0.9447	0.9212	0.9328	102
MW Organization Name	0.9213	0.9347	0.9306	43

Table 5: Fine grain Classification Report of the MWE Model using Skip-gram-LSTM where P= Precision, R= Recall and F= F-Score

The LSTM MNE model with cluster feature is able to recognize the multiword person name with 100% accuracy. The following are some of the errors encountered during testing of the model:

- The Manipuri transliterated English abbreviation were unable to be correctly recognized by the model as these words could not contribute enough to the semantic structure of the Skip-gram model from the context word.
- Most of the English multiword organization names were recognized incorrectly, which considerably decreased the accuracy, as the dataset, being a Manipuri News corpus contains fair amount of such type of words.

Overall, the model predicted the MNE tags of the words with state-of-the art accuracy considering the fact that no morphological rules or features were used in the model. Such model casts itself as break-through to Manipuri NLP computation where low resource is always a constraint.

5.4 Comparison with other MNE Classification Models

A comparative study is performed as described in Table 6. All the machine learning approaches for Manipuri MNE recognition such as SVM (Singh and Bandyopadhyay, 2010) and CRF Nongmeikapam and Bandyopadhyay (2010); Nongmeikapam et al. (2011a), use extensive morphological features such as surrounding words, word length, stemmed affixes, word counts, digit features, word frequency and NE tags.

Manipuri MWE Model	F-Score (%)
SVM (Singh and Bandyopadhyay, 2010)	MNE: 93.96%
CRF (Nongmeikapam and Bandyopadhyay, 2010)	72.24%
CRF (Nongmeikapam et al., 2011a)	92.40%
Bi-LSTM with Skip-gram Embedding	**MNE:94.98%**

Table 6: Comparison of Manipuri MWE classification Models

The F-Score measure of the proposed Manipuri MNE classification is calculated as the average of all the F-Score measures of the MNEs given in fine classification report table 5.

6 Conclusion

Finally, a deep neural model has been reported for Manipuri MNE recognition using Bi-LSTM and word embeddings in this paper. The training data consists of POS tagged words of 76526 with 12421 number of MNE. The Bi-LSTM model make use of embedding matrix generated using the Skip-gram for Manipuri Words. Word clusters information generated using the Skip-gram word vectors, is used as an additional feature. The use of K-means cluster information is to create a semantically meaningful MNE clusters. It has been reported that the cluster information generated from the Skip-gram embedding word vectors carries semantic values and has been able to supplement the MNE recognition task resulting in an increase in the average F-Score measure by 86% for MNE, as compared to the baseline model. The proposed Skip-gram model correctly represented the similarity of MNE words with about 89% accuracy. Overall the

model achieved an average F-Score measure of 94.98%.

6.1 Impact on other research

As Manipuri is a low-resource language where properly tagged dataset is unavailable, it becomes crucial for algorithms that can function without the tagged dataset. This research provides a way on how Manipuri words can be partitioned using Skip-gram embedding and K-means clustering algorithm and can provide effective solution for researches such as Manipuri News clustering, Document clustering, sentiment analysis, hate speech detection.

References

SG Ajay, M Srikanth, M Anand Kumar, and KP Soman. 2016. Word embedding models for finding semantic relationship between words in tamil language. *Indian Journal of Science and Technology*, 9(45).

Yoshua Bengio. 2012. Practical recommendations for gradient-based training of deep architectures. *CoRR*, abs/1206.5533.

Darbhe N Shankara Bhat and MS Ningomba. 1997. *Manipuri Grammar: XD-US....*, volume 4. Lincom Europa.

Paheli Bhattacharya, Pawan Goyal, and Sudeshna Sarkar. 2016. Using word embeddings for query translation for hindi to english cross language information retrieval. *Computación y Sistemas*, 20(3):435–447.

Sirajul Islam Choudhury, Leihaorambam Sarbajit Singh, Samir Borgohain, and Pradip Kumar Das. 2004. Morphological analyzer for manipuri: Design and implementation. In *Asian Applied Computing Conference*, pages 123–129. Springer.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

Yoshiaki Kitagawa and Mamoru Komachi. 2017. Long short-term memory for japanese word segmentation. *arXiv preprint arXiv:1709.08011*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Kishorjit Nongmeikapam and Sivaji Bandyopadhyay. 2010. Identification of mwes using crf in manipuri and improvement using reduplicated mwes. In *Proceedings of the International Conference on Natural Language Processing, India*.

Kishorjit Nongmeikapam and Sivaji Bandyopadhyay. 2011. Genetic algorithm (ga) in feature selection for crf based manipuri multiword expression (mwe) identification. *arXiv preprint arXiv:1111.2399*.

Kishorjit Nongmeikapam, Dhiraj Laishram, Naorem Bikramjit Singh, Ngariyanbam Mayekleima Chanu, and Sivaji Bandyopadhyay. 2011a. Identification of reduplicated multiword expressions using crf. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 41–51. Springer.

Kishorjit Nongmeikapam, Lairenlakpam Nonglenjaoba, Yumnam Nirmal, and Sivaji Bandhyopadhyay. 2011b. Improvement of crf based manipuri pos tagger by using reduplicated mwe (rmwe). *arXiv preprint arXiv:1111.2399*.

Kishorjit Nongmeikapam, Thiyam Ibungomacha Singh, Ngariyanbam Mayekleima Chanu, and Sivaji Bandyopadhyay. 2014. Manipuri chunking: An incremental model with pos and rmwe. In *Proceedings of the 11th International Conference on Natural Language Processing*, pages 277–286.

Gang Qian, Shamik Sural, Yuelong Gu, and Sakti Pramanik. 2004. Similarity between euclidean and cosine angle distance for nearest neighbor queries. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 1232–1237.

Aniketh Reddy, Monica Adusumilli, Saikiranmai Gorla, Lalita Neti, and Aruna Malapati. 2018. Named entity recognition for telugu using lstm-crf.

Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747.

Thoudam Doren Singh and Sivaji Bandyopadhyay. 2010. Web based manipuri corpus for multiword ner and reduplicated mwes identification using svm. In *Proceedings of the 1st Workshop on South and Southeast Asian Natural Language Processing*, pages 35–42.

Peilu Wang, Yao Qian, Frank K Soong, Lei He, and Hai Zhao. 2015. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *arXiv preprint arXiv:1510.06168*.

Rongchao Yin, Quan Wang, Peng Li, Rui Li, and Bin Wang. 2016. Multi-granularity chinese word embedding. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 981–986.

ScAA: A Dataset for Automated Short Answer Grading of Children's Free-text Answers in Hindi and Marathi

Dolly Agarwal[1], Somya Gupta[2], Nishant Baghel[1]

[1]Pratham Education Foundation
[2]Pratham Volunteer*
[1]{dolly.agarwal, nishant.baghel}@pratham.org, [2]somya.gupta1@gmail.com

Abstract[1]

Automatic short answer grading (ASAG) techniques are designed to automatically assess short answers written in natural language. Apart from MCQs, evaluating free text answer is essential to assess the knowledge and understanding of children in the subject. But assessing descriptive answers in low resource languages in a linguistically diverse country like India poses significant hurdles. To solve this assessment problem and advance NLP research in regional Indian languages, we present the Science Answer Assessment (ScAA) dataset of children's answers in the age group of 8-14. ScAA dataset is a 2-way (correct/incorrect) labeled dataset and contains 10,988 and 1,955 pairs of natural answers along with model answers for Hindi and Marathi respectively for 32 questions. We benchmark various state-of-the-art ASAG methods, and show the data presents a strong challenge for future research.

1 Introduction

Answer assessment is a key component of teaching and learning process and automating it has many advantages including speed, availability, consistency and fairness of assessments. Though the evaluation of multiple-choice questions is straightforward and can be scaled, there is need for systems to assess free text answers as well. Prior research has shown that recognition questions (like MCQ) are deficient as they do not capture multiple aspects of acquired knowledge such as reasoning and self-explanation (Wang et al., 2008). The free text answers are important as they help measure the understanding of the student with respect to a particular concept. Grading responses to short answer questions is considered difficult as it requires deep understanding of natural language. There can be multiple versions in which a correct answer can be articulated for the same question (examples highlighted in Table 1). Any errors in assessment can affect students' learning engagement and feedback directly.

ASAG has been in research for many years now, but most of the work has been done primarily for English. In comparison, there has been far less research in similar areas for Indian languages, which are primary medium of instruction in a large proportion of schools of India. Thus, we present a dataset for ASAG for Indian languages - Hindi and Marathi to aid development of robust solutions.

An Android app[2] for assessment was developed by our team at Pratham and was piloted in our Hybrid Learning Program. The Hybrid Learning Program of Pratham is spread across 3 states of India - Rajasthan, Uttar Pradesh and Maharashtra. The Android Assessment app enabled collection of free text answers to questions in Hindi and Marathi essential for building systems for ASAG in Indian languages. These free-text answers are 2-

* Currently affiliated with LinkedIn. This work was done as a Pratham volunteer and is not connected with LinkedIn.

[1] Data is available at: https://github.com/PrathamOrg/ScAA-Dataset

[2] Pratham Online Assessment App: https://play.google.com/store/apps/details?id=com.pratham.assessment&hl=en_IN&gl=US

Proceedings of the 17th International Conference on Natural Language Processing, pages 430–436
Patna, India, December 18 - 21, 2020.

Question in Hindi	Question in Marathi	English Translation (ETL)	Correct Answers in Hindi	English Translation	Correct Answers in Marathi	English Translation
एक वयस्क मनुष्य के कंकाल का वजन लगभग कितना होता है?	एका प्रौढ व्यक्तीच्या सांगाड्याचे वजन अंदाजे किती असते?	What is the approximate weight of an adult human skeleton?	एक वेस्ट मनुष्य के कंकाल का वजन लगभग 10 किलोग्राम होता है	The skeleton of a n adult human we ighs about 10 kil ogram	10 किग्रॅ	10 kg
			१० किग्रा	10 Kilogram	दहा किलो ग्रॅम	Ten Kilogram
			लगभग 10 किलोग्राम	Approximately 10 Kilogram		
चाँद पर कोई किसी की आवाज क्यों नहीं सुन सकते?	अंतराळवीर चंद्रावर एकमेकांचे आवाज का ऐकू शकत नाहीत?	Why can't anyone hear someone's voice on the moon?	वायुमंडल नहीं है	No atmosphere	वातावरण न सते	There is no atmosphere
			क्योंकि वहां पर वायु नहीं है	Because there is no air	तिथे हवा नाही	There is no air
			वायु मंडल का अनुपस्थिती के कारण	Because of the absence of atmosphere	कारण तिथे हवा नाही	Because there is no air
			वायुमणडल ना होने से कारण	Due to lack of air		

Table 1: Samples demonstrating that same answer can be written in multiple ways

way labeled as correct/incorrect by human annotators.

The main contributions of this paper are:

- A dataset with 10,988 answers in Hindi and 1,955 in Marathi to 32 questions in 8 different topics of science (§ 3)
- Benchmark of various state-of-the-art methods for automated assessment of free-text answers by children (§ 4)

2 Prior Art

There are numerous standard ASAG datasets publicly available for the research community to experiment with: Beetle and SciEntsBank(SEB) - released as part of SRA corpus (Dzikovska et. al, 2013), CSD (Mohler and Mihalcea 2009), X-CSD (Mohler et. al, 2011), Powergrading (PG) (Basu et. al, 2013) and ASAP (Higgins et al., 2014). The total no of prompts in these Datasets are in the range of 10 (ASAP) to 135 (SEB). While some annotated dataset (PG; Beetle) have 2-way labels (correct/incorrect), a few (CSD; X-CSD; ASAP) have scores on an ordinal scale within a range, e.g. 0-5, SEB has a more complex 5-way labels. Though there are numerous ASAG datasets available, all these are in English and none are available for Indian languages. To the best of our knowledge, this is the first comprehensive corpus with reasonable size for Hindi and Marathi.

Numerous approaches have been tried before for ASAG. Burrows et al., (2015) and Roy et al., (2015) do a comprehensive review of ASAG systems. Mohler et al. (2011) show that answers can be accurately graded by using semantic measures. Rodrigues and Araújo (2012) propose word matching between (user answer, model answer) pair. Roy et al. (2016) propose an unsupervised ASAG technique using sequential pattern mining. Sultan et al. (2016) train a supervised model based on semantic similarity features. Supervised methods include neural architectures by Riodan et al. (2017) where performance and optimal parameter settings vary across prompts, a joint-multidomain deep learning architecture by Saha et al. (2019) which learns generic and domain-specific aspects. Lun et al. (2020) introduce data augmentation strategies and show that this combined with latest BERT model brings significant gain.

For benchmarking on our dataset, we prefer unsupervised methods like sentence level semantic

similarity and sequential pattern mining due to their generalizability and suitability for deployment in an unseen-question setting. We intentionally do not benchmark supervised ML approaches that require a large corpus of labeled answers for training as they get limited to a particular question pool and are hard to generalize.

3 ScAA Data Creation

We curate Science Answer Assessment (ScAA) dataset in Hindi and Marathi language comprising of 8 science topics with 4 questions per topic, i.e. a total of 32 parallel question-model answer pairs in Hindi and Marathi. The dataset is created via three stages: Question and model answer curation, User answer collection, User answer evaluation.

The questions were selected from Grade 8 level Science topics: Adaptation, Circulatory System, Eye and Vision, Heat, Simple Machine, Skeletal System, Sound, Water Chemistry. The users here are children in the age group of 8-14 years from 3 states of India: Uttar Pradesh, Rajasthan and Maharashtra.

3.1 Data Collection and Statistics

The data is crowdsourced via an Android app developed by Pratham to enable children to take assessments anytime they want. This Android Assessment App[3] is available on play store since November 2019 with 10,000+ downloads. Children could either type the answer directly using the phone keyboard or use Speech-to-Text (STT) service[4] and then edit it. We identify the issues in the data collected via this process and pre-processing methods in section 3.2.

Over a period of 8 months, the app helped collect ~50,000 answers from 11,476 children to 32 science questions from 3 states of India. The ScAA dataset was created by selecting a subset of these answers and getting each answer evaluated as correct or incorrect by two human annotators. The Cohen's Kappa κ score indicating level of agreement between two annotators was 0.75. ScAA consists of answers where both human evaluators matched in their markings. Detailed statistics are listed in Table 2.

	Hindi	Marathi
Total Questions	32	32
# total answers	10988	1955
# total unique answers	7205	1435
# total correct answers	3843	488
Average # unique correct answers per question	41	7
Average answer length (in words)	15	15

Table 2: Statistics of Evaluated Dataset

3.2 Noise Types and Data Processing

Since any child could give the assessment whenever they like without adult supervision through phone interface, this led to presence of noise in the dataset. The option of submitting the answer through STT service brought in its own errors as well. We preprocess the noise and clean it before benchmarking. The ScAA dataset that we present lists the original noisy as well as preprocessed answers for the benefit of NLP community. Table 3 lists various noise types we found in the data and how we processed them.

Noise Type	Example	Processing
Transliterated Text	हड्डियों से appears as *Haddiyon se* in user answer	Transliterated it using Indic Transliterate*
Translated Text	हड्डियों से appears as *bones* in user answer	Translated it using Google Translate**
Code Mixed Language	bol our सॉकेट ke madat sa	Translated/Transliterated the English words
Special symbols and characters	|\£\%£`¡$°©$¶¶`=	Removed Special symbols & extra spaces
URLs	*हड्डियोंसेhttps://faq.whatsapp.com/general/26000015?lg=en&lc=IN&eea=0*	Removed these URLs
Emoji Characters	😹😹😹😹😹	Removed the emojis
Phonetically Similar words with different meanings	'ऊर्जा' (energy) recorded as 'उड जा' (fly) by STT service	Not processed

Table 3: Noise types in the data and processing

[3] Pratham Online Assessment App: https://play.google.com/store/apps/details?id=com.pratham.assessment&hl=en_IN&gl=US

[4] https://developer.android.com/reference/android/speech/package-summary

* https://pypi.org/project/indic-transliteration/

** https://pypi.org/project/googletrans/

4 Automated Short Answer Assessment

We model the assessment of user answers against reference answers as a similarity task. Each (user answer, model answer) pair is assigned a similarity score using the various state-of-the-art methods for assessment of free-text answers described in this section. We use random score assignment as baseline. User and model answers are tokenized into their constituent words using indicNLP tokenizer (Kunchukuttan et al., (2020)).

1. **Jaccard Similarity:** We calculate the number of words from user answer appearing in the model answer sentence. This is normalized w.r.t the total words present in the given answers (1), where J is the jaccard similarity score between C, the set of words in user answer and I, the set of words in model answer.

$$J = \frac{(C \cap I)}{(C \cup I)} \quad (1)$$

2. **Word based Semantic Similarity:** Answer sentences are represented by taking average of their word embeddings. We then calculate cosine similarity between them. The word embeddings used are:

   Indic NLP: Pre-trained word embeddings available for 1.1B Hindi tokens trained using FastText on corpus crawled from news websites (Kunchukuttan et al., (2020))

   fastText: Pre-trained word embeddings for Hindi, trained on Wikipedia and Common Crawl datasets consisting of 1.8B tokens (Grave et al., (2018))

3. **Sentence Similarity using S-BERT**: Sentence-BERT (Reimers and Gurevych, (2019)) finetunes a pre-trained BERT network using Siamese and triplet network and adds a pooling operation to the output of BERT to derive a sentence vector. Cosine similarity is used to compare the generated user and model answer vectors.

4. **Sequential Pattern Matching:** (Roy et. al (2016)) define a method to extract commonly occurring patterns p using support *sup(p)* to quantify the notion of commonalities from user answers and lexical diversity via type-token ratio *TTR* (eq 2). The score $Sc(s_i)$ for user answer s is calculated using this *TTR* and *sup(p)* as described in (eq 3). While this doesn't need a model answer, note that this method is most effective for batch mode as it banks on pattern mining from repeating answers and hence does not work well for real-time ASAG.

$$\text{TTR(d)} = \frac{\#distinct\ patterns\ of\ length\ d}{\#patterns\ of\ length\ d} \quad (2)$$

$$\text{Sc(s)} = \sum_{p \in si} \text{sup}\ (p)^{len(p)} * TTR(len(p)) \quad (3)$$

5 Results and Analysis

We now benchmark the ASAG methods described earlier on ScAA taking only the unique answers for evaluation (Table 4). The resulting data has 20% correct answers for Hindi and 16% for Marathi. We evaluate the models based on cost, the number of wrong assessments the similarity scores result in as compared to the actual ground truth (eq 4). We convert all scores to binary by selecting the best threshold t (table 5) for each method that minimizes this cost c and marking scores above t as 1 (correct), else 0. FP, FN, TP, TN are number of false positives, false negatives, true positives and true negatives in data.

$$c = (FP+FN) / (TP+FP+FN+TN) \quad (4)$$

Note that while on full ScAA with repeating answers PatternMatch-Repeat is comparable to S-BERT, its use is suitable in batch mode to extract answer patterns. It therefore renders itself unusable in apps, where the requirement is for real time evaluation for single (user answer, model answer) pair.

Similarity Measure	Hindi Data	Marathi Data
Baseline	0.50	0.50
Jaccard	0.76	0.75
indicNLP	0.80	0.80
fastText	0.78	0.69
S-BERT	**0.86**	**0.82**
PatternMatch-Unique	0.80	0.73
PatternMatch-Repeat	0.87	0.81

Table 4: ROC AUC for various approaches

Similarity Measure	Hindi Data	Marathi Data
Baseline	0.999	0.999
Jaccard	0.158	0.251
indicNLP	0.747	0.788
fastText	0.793	0.801
S-BERT	0.700	0.905
PatternMatch	0.245	0.539

Table 5: Threshold t for various approaches

Child's Answer / ETL	Model Answer / ETL	Human	Jaccard	indicNLP	S-BERT
मुखिया तरंग / Head waves	विद्युत् चुम्बकीय तरंग / electromagnetic waves	0	**1**	0	0
पचास किलो / Fifty Kilo	दस किग्र / Ten Kg	0	0	**1**	0
संवहन / Convection	वाष्पन / Evaporation	0	0	0	**1**
धूल के कण आंसुओं के साथ बाहर निकल आते है / Dirt particles get released with tears	धूल कण आशु के साथ में बाहर चले आते है / Dirt particles get released with tears	1	1	1	**0**
कारण तेथे वातावरण नाही / Because there's no atmosphere	कारण चंद्रावर वातावरण नाही / Because there's no atmosphere on moon	1	1	1	**0**
बल / force	बलभुजा / arm	0	0	0	**1**
प्रकाश साचे परावर्तीत / reflecting light mold	प्रकाशाचे परावर्तन / reflection of light	0	0	0	**1**

Table 6: Errors by Jaccard, indicNLP and S-BERT in marking. Columns 3,4,5,6 show marking by various methods

5.1 Error Analysis and Discussion

We now show some examples where S-BERT, IndicNLP and Jaccard based similarity measures make assessment errors in Table 6. Row 1 shows an example where the child's answer is incorrect, but Jaccard Similarity assigns it a high score due to matching word "तरंग / waves". IndicNLP incorrectly assigns high similarity among number names while S-BERT incorrectly assigns a high similarity score to word pair ("संवहन/convection", "वाष्पन/Evaporation") perhaps because they appear in similar context (rows 2 and 3).

S-BERT additionally marks a correct answer as incorrect for Hindi and Marathi in rows 4 and 5, while all the other methods mark the answer correctly. More S-BERT errors on Marathi answers are shown in rows 6 and 7. Rows 5 and 7 depict a contrast in evaluation by S-BERT where the model answer and child's answer are similar in words, but not in meaning, which it fails to capture properly. This shows its sensitivity to the input training data and absence of generalization to sentences that may unseen earlier.

The error analysis and examples showcase that while state-of-the-art models like S-BERT give best performance, they are far from being fit for deployment as the errors in assessment can directly affect students' learning engagement. Additionally, they need high latency and good compute power for assessment. A critical requirement for us is to keep the methods simple for low resource settings to cater to rural children in remote areas with limited internet access and Hindi/Marathi as the primary medium of instruction.

6 Conclusion

In this paper we present ScAA, a dataset of children's free-text answers to 32 questions of grade 8 level Science topics in Hindi and Marathi along with their user answers. This dataset is intended to facilitate research in automatic assessment of short answers in Indian languages. We benchmark the performance of various state-of-the-art ASAG methods on ScAA and observe that even though BERT based model performs best, it makes errors in assessment that can affect students' learning engagement, thus leaving scope for improvement before such techniques can be deployed in real world and presenting a strong case for more research in this area. We believe that this dataset will be useful for the research community working on automated short answer assessment for Indian languages and aid in solving a very practical problem for society at scale.

References

Basu, Sumit, Chuck Jacobs, and Lucy Vanderwende. "Powergrading: a clustering approach to amplify human effort for short answer grading." *Transactions of the Association for Computational Linguistics* 1 (2013): 391-402.

Burrows, Steven, Iryna Gurevych, and Benno Stein. "The eras and trends of automatic short answer grading." *International Journal of Artificial Intelligence in Education* 25, no. 1 (2015): 60-117.

Dzikovska, Myroslava O., Rodney D. Nielsen, Chris Brew, Claudia Leacock, Danilo Giampiccolo, Luisa Bentivogli, Peter Clark, Ido Dagan, and Hoa Trang Dang. "SemEval-2013 Task 7: The Joint Student

Response Analysis and 8th Recognizing Textual Embodiment Challenge." In *Second Joint Conference on Lexical and Computational Semantics (* SEM): Seventh International Workshop on Semantic Evaluation (SemEval 2013), vol. 2. Association for Computational Linguistics*, 2013.

E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov. 2018. Learning Word Vectors for 157 Languages. *In Proceedings of LREC 2018, 15th conference on International Language Resources and Evaluation*.

Hao-Chuan Wang, Chun-Yen Chang, and Tsai-Yen Li. 2008. Assessing Creative Problem-solving with Automated Text Grading. *Computers and Education*, 51(4):1450–1466.

Higgins, Derrick, Chris Brew, Michael Heilman, Ramon Ziai, Lei Chen, Aoife Cahill, Michael Flor et al. "Is getting the right answer just about choosing the right words? The role of syntactically-informed features in short answer scoring." *arXiv preprint* arXiv:1403.0801 (2014).

Horbach, Andrea, and Manfred Pinkal. "Semi-supervised clustering for short answer scoring." In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. 2018.

Kunchukuttan, A., Kakwani, D., Golla, S., Bhattacharyya, A., Khapra, M.M. and Kumar, P., 2020. AI4Bharat-IndicNLP Corpus: Monolingual Corpora and Word Embeddings for Indic Languages. *arXiv preprint* arXiv:2005.00085.

Kumar, Sachin, Soumen Chakrabarti, and Shourya Roy. "Earth Mover's Distance Pooling over Siamese LSTMs for Automatic Short Answer Grading." In *IJCAI*, pp. 2046-2052. 2017.

Lun, Jiaqi, Jia Zhu, Yong Tang, and Min Yang. "Multiple Data Augmentation Strategies for Improving Performance on Automatic Short Answer Scoring." In *AAAI*, pp. 13389-13396. 2020.

Mieskes, Margot, and Ulrike Pado. "Work Smart-Reducing Effort in Short-Answer Grading." In *Proceedings of the 7th Workshop on NLP for Computer Assisted Language Learning (NLP4CALL 2018) at SLTC, Stockholm, 7th November 2018*, no. 152, pp. 57-68. Linköping University Electronic Press, 2018.

Mohler, Michael, and Rada Mihalcea. "Text-to-text semantic similarity for automatic short answer grading." In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pp. 567-575. 2009.

Mohler, Michael, Razvan Bunescu, and Rada Mihalcea. "Learning to grade short answer questions using semantic similarity measures and dependency graph alignments." *In Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pp. 752-762. 2011.

Pérez, Diana, Alfio Massimiliano Gliozzo, Carlo Strapparava, Enrique Alfonseca, Pilar Rodríguez, and Bernardo Magnini. "Automatic Assessment of Students' Free-Text Answers Underpinned by the Combination of a BLEU-Inspired Algorithm and Latent Semantic Analysis." In *FLAIRS conference*, pp. 358-363. 2005.

Piyush Patil, Sachin Patil, Vaibhav Miniyar and Amol Bandal.2018. Subjective Answer Evaluation Using Machine Learning in *International Journal of Pure and Applied Mathematics*, Volume 118 No. 24 2018

Ramachandran, Lakshmi, Jian Cheng, and Peter Foltz. "Identifying patterns for short answer scoring using graph-based lexico-semantic text matching." In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pp. 97-106. 2015.

Rodrigues, Fátima & Araújo, Lília. (2012). Automatic Assessment of Short Free Text Answers. *4th International Conference on Computer Supported Education*. 2.

Reimers, Nils, and Iryna Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks." *arXiv preprint* arXiv:1908.10084 (2019).

Riordan, Brian, Andrea Horbach, Aoife Cahill, Torsten Zesch, and Chungmin Lee. "Investigating neural architectures for short answer scoring." In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pp. 159-168. 2017.

Roy, Shourya, Sandipan Dandapat, Ajay Nagesh, and Yadati Narahari. "Wisdom of students: A consistent automatic short answer grading technique." In *Proceedings of the 13th International Conference on Natural Language Processing*, pp. 178-187. 2016.

Roy, Shourya, Y. Narahari, and Om D. Deshmukh. 2015. A Perspective on Computer Assisted Assessment Techniques for Short Free-Text Answers. *In Proceedings of the International Conference on Computer Assisted Assessment (CAA)*, pages 96–109. Springer.

Sultan, Md Arafat, Cristobal Salazar, and Tamara Sumner. "Fast and easy short answer grading with high accuracy." In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1070-1075. 2016.

Sahu, Archana, and Plaban Kumar Bhowmick. "Feature Engineering and Ensemble-Based Approach for Improving Automatic Short-Answer Grading Performance." *IEEE Transactions on Learning Technologies* 13, no. 1 (2019): 77-90.

Saha, Swarnadeep, Tejas I. Dhamecha, Smit Marvaniya, Peter Foltz, Renuka Sindhgatta, and Bikram Sengupta. "Joint Multi-Domain Learning for Automatic Short Answer Grading." *arXiv preprint* arXiv:1902.09183 (2019).

Wang, Tianqi, Tomoya Mizumoto, Naoya Inoue, and Kentaro Inui. "Identifying Current Issues in Short Answer Grading." *ANLP-2018* (2018).

Zhang, Yuan, Rajat Shah, and Min Chi. "Deep Learning+ Student Modeling+ Clustering: A Recipe for Effective Automatic Short Answer Grading." *International Educational Data Mining Society* (2016).

Exploring Pair-Wise NMT for Indian Languages

Kartheek Akella[*]
CVIT, IIIT-H
sukruthkartheek@gmail.com

Sai Himal Allu[*]
CVIT, IIIT-H
saihimal.allu@gmail.com

Sridhar Suresh Ragupathi[*]
CVIT, IIIT-H
srsridhar.98@gmail.com

Aman Singhal
CVIT, IIIT-H
amansinghalml@gmail.com

Zeeshan Khan
CVIT, IIIT-H
zeeshank606@gmail.com

Vinay P. Namboodiri
University of Bath
vpn22@bath.ac.uk

C V Jawahar
CVIT, IIIT-H
jawahar@iiit.ac.in

Abstract

In this paper, we address the task of improving pair-wise machine translation for specific low resource Indian languages. Multilingual NMT models have demonstrated a reasonable amount of effectiveness on resource-poor languages. In this work, we show that the performance of these models can be significantly improved upon by using back-translation through a filtered back-translation process and subsequent fine-tuning on the limited pair-wise language corpora. The analysis in this paper suggests that this method can significantly improve a multilingual models' performance over its baseline, yielding state-of-the-art results for various Indian languages.

1 Introduction

Neural machine translation (NMT) algorithms, as is common for most deep learning techniques, work best with vast amounts of data. Various authors have argued that their performance would be limited for low resource languages (Östling and Tiedemann, 2017), (Gu et al., 2018), (Kim et al., 2020). One way to bridge this gap is through the use of multilingual NMT algorithms to bypass the data limitations of individual language pairs ((Johnson et al., 2017), (Aharoni et al., 2019), (Vázquez et al., 2019)). The use of such a model has been demonstrated recently by (Philip et al., 2021). In this paper, we investigate the problem of improving pair-wise NMT performance further over existing multilingual baselines. We specifically analyze the use of back-translation and fine-tuning to this effect. Our results suggest that it is possible to improve the performance of individual pairs of languages for various Indian languages. The performance of these language pairs is evaluated over standard datasets, and we observe consistent improvements over the two main baselines: an NMT system trained pair-wise from scratch using the corpora available for the pair of languages, a multilingual NMT model that uses many different languages.

2 Previous Work

The problem of multilingual NMT has attracted significant research attention in the recent past. (Dong et al., 2015) proposed the first multilingual model with a one-to-many mapping of languages, whereas (Ferreira et al., 2016) shared a single attention network for all language pairs. Recent works like (Conneau and Lample, 2019), (Conneau et al., 2020) which are an extension of (Liu et al., 2019) and (Devlin et al., 2019) have improved upon the initial formulation of the multilingual NMT problem. Works like (Currey et al., 2017), (Shah and Barber, 2018), (Li and Eisner, 2019) and (Hewitt and Liang, 2019) use monolingual data to supplement their parallel corpora to build an NMT system. On the flip side (Lample et al., 2018), (Wang et al., 2018), (Artetxe et al., 2018) study the unsupervised paradigm using only monolingual corpora.

Within the context of Indian languages, (Chandola and Mahalanobis, 1994) and (Dave

*Equal Contribution. Sridhar worked on Tamil and Urdu, Himal worked on Gujarati, Kartheek worked on Malayalam and Marathi, Aman worked on Hindi and Punjabi and Zeeshan worked on Odiya. Himal was responsible for drafting this paper.

Proceedings of the 17th International Conference on Natural Language Processing, pages 437–443
Patna, India, December 18 - 21, 2020.

et al., 2001) were one of the first works to explore a rule-based approach for translation from Hindi to English whereas (Patel et al., 2018), (Barman et al., 2014), (Saini and Sahula, 2018) and (Choudhary et al., 2018) have explored this problem through the prism of NMT. (Philip et al., 2019) and (Madaan and Sadat, 2020) extend the concept of multilingual NMT to the setting of Indian languages. Due to the recent efforts undertaken by the authors of Kakwani et al. (2020), Indian languages are now better represented in terms of available monolingual corpora. These resources set up a fertile ground for exploitation by semi-supervised and unsupervised NMT approaches, which are consistent with the setting we study in this work.

3 Method

Consider a setting in which we have limited pair-wise corpora between a pair of languages, and we would like to obtain improved performance. We go about achieving this through the following procedure. First, we train a multilingual model on several languages. Next, we use existing monolingual corpora through back-translation (BT), and then we fine-tune the model using available pair-wise corpora. We show that this particular procedure indeed improves over the alternative approach of training a pair-wise NMT system using the available corpora. For the first step, we use the multilingual NMT model provided publicly by (Philip et al., 2021). We now provide details regarding the other two stages.

3.1 Back Translation

In NMT literature, BT is an effective approach that allows NMT to pivot from a fully supervised setting to a semi-supervised setting. When supplemented with other objectives (autoencoder denoising (Artetxe et al., 2018), cross-translation (García et al., 2020)), BT has demonstrated high efficacy in a fully unsupervised NMT setting as well. In the semi-supervised paradigm, which we study in this work, our object of interest is to generate a meaningful learning signal from monolingual resources of a particular language that allows a reasonable NMT model to exploit these resources to improve its performance on language pairs which include that specific language.

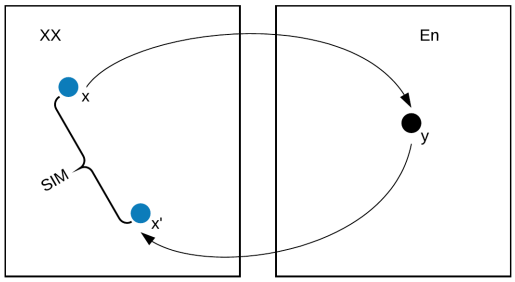


Figure 1: SIM stands for a similarity heuristic. We use sentence wise BLEU scores in our work

Filtering Mechanism

Using a reasonable NMT model, BT can leverage monolingual resources to generate notable amounts of low-quality synthetic parallel data. If low-quality parallel corpora can be filtered through some means such that erroneous translation pairs are eliminated, we can obtain a strong learning signal from such a filtered corpus. To design such a filtering mechanism, we draw inspiration from generative modelling literature, precisely the idea of cyclic consistency (Zhu et al., 2017). Briefly, the idea of cyclical consistency within the context of computer vision relates to minimizing the discrepancy between an image from domain X and the image obtained after transforming it to a domain Y and then converting it back to the domain X. We adopt this approach to build our filtering mechanism, in which we first use our reasonable NMT model to generate intermediate English (EN) translations for the sentences in the monolingual corpus of some language (XX). As illustrated in fig 1, we then use these intermediate English translations to back-translate it into XX and then evaluate the sentence wise BLEU (Papineni et al., 2002) scores for each such translation as a measure of similarity. Only sentences that cross an empirically chosen threshold are retained to ensure that the generated translations are of good quality to obtain a reasonably high-quality synthetic parallel dataset. We refer to this filtering scheme as XX-EN-XX.

We initialise our NMT model using the weights from the multilingual NMT model pro-

	Pre-filt #pairs	Post-filt #pairs
hi	4M	140K
pa	58K	7K
mr	178K	58K
gu	370K	39K
ta	88K	34K
ur	400K	105K
ml	178K	52K
od	221K	64K

Table 1: Monolingual corpora utilized

	hi	pa	gu	mr	ta	ml	ur	od
iitb	1.5M	-	-	-	-	-	-	-
cvit-pib	195K	27K	29K	81K	87K	32K	45K	-
ufal	-	-	-	-	167K	-	-	-
ilci	49K	49K	49K	-	49K	30K	49K	-
odcorp1.0	-	-	-	-	-	-	-	27K
odcorp2.0	-	-	-	-	-	-	-	97K
	1.75M	76K	79K	81K	303K	62K	94K	124K

Table 2: Parallel corpora utilized

vided by the authors of (Philip et al., 2021). The authors train a Transformer (Vaswani et al., 2017) model on 10 Indian languages namely (notation in brackets), Hindi (hi), Telugu (te), Tamil (ta), Malayalam (ml), Urdu (ur), Bangla (bn), Gujarati (gu), Marathi (mr) and Odia (od) in addition to English (en). They make two chief architectural decisions in this regard. One they develop a shared vocabulary over all languages of interest, giving equal representation to each language in the vocabulary (equal number of tokens from each language). The second, they share the encoder-decoder parameters of the Transformer model across all possible language pairs, a decision which encourages the model to learn a shared embedding space for all languages of interest. The low resource nature of these languages is primarily addressed through two techniques: namely, Transfer Learning and Backtranslation (Sennrich et al., 2016). This design choice allows us to use the same NMT model for the XX-EN and the EN-XX directions of the XX-EN-XX setting with a consistent amount of effectiveness. We reason that an EN-XX-EN filtering scheme would result in a compounding of errors problem due to the superior performances offered by the multilingual NMT model in the XX-EN direction in contrast to the EN-XX direction. In such a case, populating our filtered corpus would require selecting a lower value of the threshold, which would compromise the quality of translation pairs, thereby leading to a weak supervisory signal.

4 Experimental Setup

4.1 Training Details

We make use of the Transformer-Base, a part of Fairseq library (Ott et al., 2019), which is built with 6 encoder-decoder layers, each having 512 hidden units and a singular attention head as our NMT model. We initialise our model with the weights of the multilingual NMT model provided by the authors of (Philip et al., 2021). We also utilise the SentencePiece (Kudo and Richardson, 2018) models of (Philip et al., 2021) to build our vocabulary.

For all languages of interest, we carry out filtering of the back-translated corpus by first evaluating the mean of sentence-wise BLEU scores for the cyclically generated translations and then selecting a value slightly higher than the mean as our threshold. Sentences that cross this threshold are then included along with their corresponding translations in our filtered corpus. We supplement the training of our NMT model on a filtered back-translated corpus with two rounds of finetuning on a relevant parallel corpus: a pre-training phase and a post-training phase before carrying out the final evaluation. We reason that a pre-training step enhances the possibility of generating a high-quality synthetic filtered corpus from the related monolingual corpora by providing a more robust prior NMT model for the BT routine. A post-training step ensures that the NMT model is subjected to a more reliable supervisory signal before the final evaluation is carried out. We train all our models using AdamW (Loshchilov and Hutter, 2019) optimizer until a local minimum is achieved.

4.2 Datasets

For Hi, Od, and Ta, we use the IIT-B corpus, OdiEnCorp-v1.0, and PMIndia Corpus (Haddow and Kirefu, 2020) respectively. For the rest of the languages, we utilise the relevant monolingual corpora provided by the authors of (Kakwani et al., 2020). We use only

		State of the Art				NMT (Different Attempts)		
PAIRS	Test-Set	Top-4 (Prev. Attempts)				Rand-init	M-NMT[1]	Filt-BT
En-Hi	MKB	15.65[2]	16.23[2]	21.05[2]	**24.48**[2]	13.28	16.93	16.67
En-Pa	ILCI				23.05[1]	10.67	21.36	**23.52**
En-Mr	MKB	8.79[2]	8.84[2]	8.97[2]	9.65[2]	2.77	9.84	**9.89**
En-Gu	MKB	9.73[2]	10.13[2]	11.24[2]	11.70[2]	2.63	12.92	**14.37**
En-Ta	MKB	4.33[2]	4.43[2]	4.53[2]	4.94[2]	0.78	4.86	**5.69**
En-Ta	UFAL	11.73[2]	12.51[2]	12.74[2]	13.05[2]	0.78	7.80	**19.07**
En-Ml	MKB	5.00[2]	5.17[2]	5.42[2]	6.32[2]	1.59	2.65	**6.40**
En-Ur	MKB				22.16[1]	3.90	22.16	**24.76**
En-Od	ODIENCORPv2	7.93[2]	9.35[2]	9.85[2]	**11.07**[2]	5.29	0.96	10.84

Table 3: Comparison of our NMT results with others publicly available on WAT leader board [2]. For results that were not available on WAT leaderboard (Pa,Ur), we compare it with results from the paper (Philip et al., 2021). We find that initialisation using a multlilingual model[1] is highly effective for NMT in contrast to initialising randomly and training only on the respective language

a part of these monolingual corpora in our experiments, the statistics of which we present in Table 1. Our pre-training and post-training routines dictate the need for a parallel corpus for all our languages of interest. Due to the relative lack of availability of large high-quality parallel corpora for our language pairs of interest, we collate available resources to obtain a final parallel corpus, the details of which are presented in Table 2. In addition to CVIT-PIB (Siripragada et al., 2020) and ILCI (Jha, 2010) datasets, we also utilise IIT-B Hi-En corpus (Kunchukuttan et al., 2018) for Hi, UFAL EnTamv2.0 (Ramasamy et al., 2012) for Ta and OdiEnCorp 1.0 and 2.0 (Parida et al., 2020) for Od. For evaluation we use the CVIT-MKB (Siripragada et al., 2020) dataset for the languages in MKB. We evaluate on the ILCI dataset for Pa, OdiEncorp-v2.0 (Parida et al., 2020) for Od and UFAL EnTamv2.0 (Ramasamy et al., 2012) for Tamil.

5 Results and Discussions

We report BLEU scores on all the test sets specified. We refer to our approach as Filt-BT in Table 3 and contrast our results with a randomly initialised model trained from scratch with the same conditions (Rand-Init), the multilingual model that we use as our prior NMT model (M-NMT) [1] and the top 4 publicly available results on the WAT leaderboard [2] [3]. Since Pa and Ur do not have an entry on the leaderboard, we instead make a comparison with the results reported in (Philip et al., 2021), which are the present SOTA results to the best of our knowledge.

The first comparison highlights the benefits of warm-starting our NMT model from a M-NMT model, whereas the second comparison helps us ascertain the efficacy of filtered BT and as such, we report consistent gains over both these baselines for all the language pairs. In all the language pairs barring Odia, we demonstrate the superior performances of a prior multilingual model in contrast to a specialized model trained from scratch, validating our claim that initialization using a multilingual model is highly effective for NMT in contrast to initializing randomly and training only on the respective language. Typically, we observe that using high threshold values for filtering leads to the filtered corpus getting biased by selecting comparatively shorter sentences. To maintain a healthy mix of both types of sentences, we use a threshold value slightly higher than the mean of the sentence-wise BLEU scores which we find in our experiments empirically provides for a more balanced high quality (in terms of translation quality) corpus thereby guaranteeing a better supervisory signal.

[1] Philip et al. (2021)

[2] http://lotus.kuee.kyoto-u.ac.jp/WAT/WAT2020/

[3] We do not include results from model-ensembling approaches

For Ta and Ur, we notice a massive boosts in performance (11.88 and 10.49 BLEU points respectively) over our multilingual baseline, with significant gains, also being noticed for Ml and Gu. The M-NMT model, which we use to initialize our NMT model, has been trained on OdiEnCorpv1.0, whereas our Rand-Init model has been trained using both versions of the dataset. We ascribe the former's inferior performance compared to the latter on Odia to the domain mismatch between both these versions, something which the latter model does not have to face.

We select a subset of the monolingual data to maintain consistency with our computing resources. For 200K sentences, we train on a 1080ti NVIDIA GPU and found that back translation took about 3 hours. We decided that it would be an adequate sample size to test the validity of our approach. Since only a subset of monolingual data provided by (Kakwani et al., 2020) is used, we fully expect these results to trend upwards if the entire corpora were to be utilised. We indicate the SOTA performance for each language in bold. As such, we achieve SOTA performances on Pa, Gu, Ml, Mr, Ta and Ur.

6 Summary and Directions

Our explorations in the applicability of Neural Machine Translation for Indian languages lead to the following observations (i) Multilingual models are a promising direction to address data scarcity and the variability of resources across languages (ii) adapting a multilingual model for a specific pair can provide superior performances. We believe these solutions can further benefit from the availability of monolingual resources and noisy parallel corpora.

References

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3874–3884, Minneapolis, Minnesota. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *Proceedings of the Sixth International Conference on Learning Representations*.

Anup Barman, Jumi Sarmah, and Shikhar Sarma. 2014. Assamese WordNet based quality enhancement of bilingual machine translation system. In *Proceedings of the Seventh Global Wordnet Conference*, pages 256–261, Tartu, Estonia. University of Tartu Press.

Anoop Chandola and Abhijit Mahalanobis. 1994. Ordered rules for full sentence translation: A neural network realization and a case study for hindi and english. *Pattern Recognit.*, 27:515–521.

Himanshu Choudhary, Aditya Kumar Pathak, Rajiv Ratan Saha, and Ponnurangam Kumaraguru. 2018. Neural machine translation for English-Tamil. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 770–775, Belgium, Brussels. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 7059–7069. Curran Associates, Inc.

Anna Currey, Antonio Valerio Miceli Barone, and Kenneth Heafield. 2017. Copied monolingual data improves low-resource neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 148–156, Copenhagen, Denmark. Association for Computational Linguistics.

Shachi Dave, Jignashu Parikh, and Pushpak Bhattacharyya. 2001. Interlingua-based english–hindi machine translation and language divergence. *Machine Translation*, 16(4):251–304.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. 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 1723–1732, Beijing, China. Association for Computational Linguistics.

Daniel C. Ferreira, André F. T. Martins, and Mariana S. C. Almeida. 2016. Jointly learning to embed and predict with multiple languages. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2019–2028, Berlin, Germany. Association for Computational Linguistics.

X. García, P. Forêt, Thibault Sellam, and Ankur P. Parikh. 2020. A multilingual view of unsupervised machine translation. *ArXiv*, abs/2002.02955.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O.K. Li. 2018. Universal neural machine translation for extremely low resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 344–354, New Orleans, Louisiana. Association for Computational Linguistics.

Barry Haddow and Faheem Kirefu. 2020. Pmindia – a collection of parallel corpora of languages of india.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

Girish Nath Jha. 2010. The TDIL program and the Indian langauge corpora intitiative (ILCI). In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Languages Resources Association (ELRA).

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. 2020. IndicNLPSuite: Monolingual Corpora, Evaluation Benchmarks and Pre-trained Multilingual Language Models for Indian Languages. In *Findings of EMNLP*.

Yunsu Kim, Miguel Graça, and Hermann Ney. 2020. When and why is unsupervised neural machine translation useless? In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 35–44, Lisboa, Portugal. European Association for Machine Translation.

Taku Kudo and J. Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *EMNLP*.

Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. 2018. The IIT Bombay English-Hindi parallel corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, Miyazaki, Japan. European Languages Resources Association (ELRA).

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations (ICLR)*.

Xiang Lisa Li and Jason Eisner. 2019. Specializing word embeddings (for parsing) by information bottleneck. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2744–2754, Hong Kong, China. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization.

Pulkit Madaan and Fatiha Sadat. 2020. Multilingual neural machine translation involving Indian languages. In *Proceedings of the WILDRE5–5th Workshop on Indian Language Data: Resources and Evaluation*, pages 29–32, Marseille, France. European Language Resources Association (ELRA).

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Shantipriya Parida, Satya Ranjan Dash, Ondřej Bojar, Petr Motlicek, Priyanka Pattnaik, and Debasish Kumar Mallick. 2020. OdiEnCorp 2.0: Odia-English parallel corpus for machine translation. In *Proceedings of the WILDRE5–5th Workshop on Indian Language Data: Resources and Evaluation*, pages 14–19, Marseille, France. European Language Resources Association (ELRA).

Raj Patel, Prakash Pimpale, and Sasikumar Mukundan. 2018. Machine translation in indian languages: Challenges and resolution. *Journal of Intelligent Systems*.

Jerin Philip, Vinay P. Namboodiri, and C. V. Jawahar. 2019. A baseline neural machine translation system for indian languages.

Jerin Philip, Shashank Siripragada, Vinay P. Namboodiri, and C. V. Jawahar. 2021. Revisiting low resource status of indian languages in machine translation (pre-print available on arxiv). In *Proceedings of ACM India Joint International Conference on Data Science Management of Data*, Bangalore, India.

Loganathan Ramasamy, Ondřej Bojar, and Zdeněk Žabokrtský. 2012. Morphological processing for english-tamil statistical machine translation. In *Proceedings of the Workshop on Machine Translation and Parsing in Indian Languages (MTPIL-2012)*, pages 113–122.

S. Saini and V. Sahula. 2018. Neural machine translation for english to hindi. In *2018 Fourth International Conference on Information Retrieval and Knowledge Management (CAMP)*, pages 1–6.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Harshil Shah and David Barber. 2018. Generative neural machine translation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 1346–1355. Curran Associates, Inc.

Shashank Siripragada, Jerin Philip, Vinay P. Namboodiri, and C V Jawahar. 2020. A multilingual parallel corpora collection effort for Indian languages. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 3743–3751, Marseille, France. European Language Resources Association.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Raúl Vázquez, Alessandro Raganato, Jörg Tiedemann, and Mathias Creutz. 2019. Multilingual NMT with a language-independent attention bridge. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 33–39, Florence, Italy. Association for Computational Linguistics.

Wei Wang, Taro Watanabe, Macduff Hughes, Tetsuji Nakagawa, and Ciprian Chelba. 2018. Denoising neural machine translation training with trusted data and online data selection. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 133–143, Belgium, Brussels. Association for Computational Linguistics.

Jun-Yan Zhu, T. Park, Phillip Isola, and A. Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251.

Robert Östling and Jörg Tiedemann. 2017. Neural machine translation for low-resource languages.

Only text? only image? or both? Predicting sentiment of internet memes

Pranati Behera, Mamta, Asif Ekbal
Department of Computer Science and Engineering
Indian Institute of Technology Patna, India
pranati5079@gmail.com, {mamta_1921cs11, asif}@iitp.ac.in

Abstract

Nowadays, the spread of Internet memes on online social media platforms such as Instagram, Facebook, Reddit, and Twitter is very fast. Analyzing the sentiment of memes can provide various useful insights. Meme sentiment classification is a new area of research that is not explored yet. Recently SemEval provides a dataset for meme sentiment classification. As this dataset is highly imbalanced, we extend this dataset by annotating new instances and use a sampling strategy to build a meme sentiment classifier. We propose a multi-modal framework for meme sentiment classification by utilizing textual and visual features of the meme. We found that for meme sentiment classification, only textual or only visual features are not sufficient. Our proposed framework utilizes textual as well as visual features together. We propose to use the attention mechanism to improve meme classification performance. Our proposed framework achieves macro F1 and accuracy of 34.23% and 50.02%, respectively. It increases the accuracy by 6.77% and 7.86% compared to only textual and visual features, respectively.

1 Introduction

The rapid growth of users on social media platforms leads to new ways of spreading information. Meme nowadays has become one of the most popular words for social media. A meme is an idea, the way in which a person behaves in response to a particular situation or a manner that spreads from one person to another within a culture. Spreading of memes on social media platforms such as Facebook, Instagram, Reddit, and Twitter is very fast.

Sentiment analysis is a growing field of Natural Language Processing (NLP), aiming to identify the polarity of opinion. Sentiment can be positive, negative or neutral (Pang and Lee, 2005). Sentiment analysis has a vast number of applications in real life, including the product's recommendation to a user based on opinions provided by other users (Pang et al., 2002), in political uses (Bakliwal et al., 2013), etc. Memes play an important role in handling various political battles or public relations on social media platforms.

The most common practice in sentiment analysis is finding the sentiment of textual content crawled from Twitter, product reviews, hotel reviews, etc. Existing literature has mostly addressed the problem of sentiment analysis primarily using textual contents (Xu et al., 2019; Edara et al., 2019; Medhat et al., 2014; . et al., 2020). But with the growing social media, users are expressing their opinions through text and the image. Hence, researchers nowadays are also giving attention to sentiment analysis in multi-modal content (You et al., 2016; Ortis et al., 2020; Man et al., 2019). Spreading of memes is also very fast, but meme analysis is yet to be explored. Recently, SemEval-2020 proposed a task to detect the meme's polarity, which can fall into three predefined classes: positive, negative, or neutral (Sharma et al., 2020). This is the very first attempt towards the meme sentiment analysis.

To analyze the sentiment of memes, the text-only approaches may not be sufficient. For example, consider the meme given in Figure 1, if the only textual content is considered, then the sentence 'FINALLY GETS JOB INTERVIEW' seems to have a neutral sentiment (no explicit positive words are used). However, if we also consider the visual information of meme, as shown in Figure 1, then we can say overall sentiment is positive. Hence, to analyze memes, both text and visual features have their own importance.

In this paper, we work on the SemEval-2020 Task-8 dataset to detect the sentiment of memes. But this dataset is imbalanced. Hence we extend this dataset by adding more training instances for

Proceedings of the 17th International Conference on Natural Language Processing, pages 444–452
Patna, India, December 18 - 21, 2020.

Figure 1: Meme example

balancing purposes and then propose a multi-modal framework based on deep neural networks to classify the sentiment of the meme into one of the predefined classes, namely positive, negative, and neutral. We use a multi-modal framework with attention applied to both image and text to find out important regions and important words. Thereafter, to combine the image and textual modality, we use a fully connected layer that tries to find the relation between textual and visual features and finally produces a combined feature vector. We evaluate the proposed approach using accuracy and Macro F1 score on the test set of the SemEval-2020 dataset. We get the macro F1 of 34.23%, and accuracy of 50.02%, respectively, which is higher than the SemEval baseline, i.e., Macro F1 of 0.21%.

The rest of the paper is organized as follows. Section 2 describes the related work. Section 3 describes the methodology for classification. Section 4 describes the data collection, annotation, and experimental setup. Section 5 describes the results and detailed error analysis. Section 6 concludes the paper and describes future research plans.

2 Related work

This section describes the works on sentiment analysis for text as well as for multi-modal content. (Murty and Allu) proposed an approach for finding the sentiment analysis on text reviews by using Long Short Term Memory. (Agarwal et al., 2011) proposed the framework to classify the sentiment of tweets into positive, negative and neutral class using prior polarity scoring, which is based on the prior polarity of words. (Li et al., 2019) proposed a sentiment-feature-enhanced deep neural network (SDNN) to detect the sentiment of text by deep neural network integrated with sentiment linguistic knowledge via attention mechanism. (Mozetič et al., 2016) proposed a framework for textual sentiment analysis using lexicon based and machine learning based approach. Sentiment is predicted from the set of sentiment-bearing words identified in the text using lexicons. (Ghiassi and Lee, 2018) proposed a set of domain transferable Twitter lexicons, obtained from tweets for the task of sentiment analysis. (Kumar and Jaiswal, 2017) proposed a model to detect the sentiment of images using Convolutional neural network. They used Flicker images dataset to train their model and Twitter images dataset for testing. (Akhtar et al., 2020) proposed a stacked ensemble model for predicting the degree of intensity for sentiment and emotion. They used multi-layer perceptron network to combine outputs of feature based models and deep learning models. (Poria et al., 2018) explored different deep-learning based architectures for multi-modal sentiment classification. They used deep convolutional neural network (CNN) to extract features from the visual and text modalities. (Jiang et al., 2020) proposed a fusion-extraction network model for multi-modal sentiment analysis. Their proposed model learned two types of representations, visual-specific textual representations and textual-specific visual representations using interactive information fusion mechanism.

Above mentioned works are either for text or multi-modal content. Meme classification has not been explored much in detail. So we proposed a framework for meme classification by utilizing text written on it and image features.

3 Methodology

This section represents our proposed methodology in detail. We develop a multi-modal neural network that learns from the two modalities, *viz.* textual and visual. For text modality, our model takes as input the embedding representation of each word present in the OCR extracted text. Further, we use Convolutional Neural Network (CNN) to learn textual features, and then we apply attention to the output of CNN to extract the most relevant features for classification. We use the pretrained model VGGNet to extract the visual features for image modality, and then we apply attention to the extracted features to detect important visual features for classification. Finally, both the features

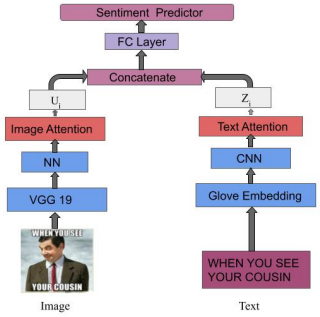


Figure 2: Proposed architecture

are fused with the help of a fully connected layer. The overall architecture of our proposed model is shown in Figure 2.

3.1 Textual features

In this section, we discuss the textual features, how they are given as input to our system, and how to apply attention to the features extracted.

3.1.1 Embedding Layer

The embedding layer takes the input as a sequence of words present in the sentence. For each word w present in the sentence, a lookup matrix is created to obtain its embedding representation. Lookup matrix can be initialized using pretrained word embedding vectors (Bojanowski et al.; Pennington et al., 2014). In our work, the pre-trained vector representations provided by Glove (Pennington et al., 2014) are used. It captures syntactic and semantic relations among the words. The embedding of each word w is then given as an input to the CNN to learn the text representation. Equation 1 shows the sequence of words present in sentence where w_i is i_{th} word present in the sentence and L is length of sentence.

$$W_i = w_i^1, w_i^2, ...w_i^j, ...w_i^L \tag{1}$$

3.1.2 Convolutional Neural Network (CNN)

The convolutional neural network automatically learns the features with the help of convolutional filters. Convolutional filters capture the semantic and syntactic features of a given sentence. CNN has been used in a wide variety of tasks (Rios and Kavuluru, 2015), (Kim, 2014). The CNN consists of convolutional layers. Convolutional layers are followed by non-linear layers that contain the Relu activation function, followed by the pooling layers. For our task, we use 3 convolutional layers. The three convolutional layers contain 128 filters of sizes 2, 3, and 4 each. Word embedding vectors of a sentence are given as input to CNN to learn the n-gram features. Equation 2 shows the CNN output for a sentence after convolving different size filters on the word embedding matrix of the sentence.

$$H_i = h_i^1, h_i^2, ...h_i^j, ...h_i^L \tag{2}$$

Where H_i represents the final feature vector for a sentence.

3.1.3 Attention for text

In NLP related tasks, some words in the sentence are more important for the task compared to the other words in the same sentence. To capture this phenomena, attention model for the text has been proven beneficial for many NLP related tasks i.e., text summarization, machine translation (Luong et al., 2015; Bahdanau et al., 2014), textual sentiment analysis(Corpora, 2000; Chen et al., 2016), etc. Attention models calculate the attention score α_i^j which lies in the range of 0 and 1. Attention score is assigned to feature representation of each w_i^j i.e., h_i^j based on its importance, which is calculated as follows

$$\alpha_i^j = \frac{exp(p_i^j)}{\sum_{j=1}^{L} exp(p_i^j)} \tag{3}$$

Where,

$$p_i^j = \theta(Mh_i^j + b) \tag{4}$$

θ refers to nonlinear activation function ($tanh$). The weight matrix M and bias b are the network parameters and h_i^j is the feature representation of word w_i^j (CNN output). α is calculated for all the words in the sentence. The attended text feature vector can be calculated as a weighted sum of all the words present in a sentence, as shown in Equation 5.

$$Z_i^k = \sum_{1<=j<=L} \alpha_i^j h_i^j \tag{5}$$

Attention process for text is illustrated Figure 3.

3.2 Visual Features

The image with size 224*224 is used as the input to the pre-trained model VGG-19 to extract features of the image. We use the output of *conv5*4* layer of VGG-19 as the region features which consist of 196 regions, and each region is represented in 512 dimensions. Thus region features are having dimensions of (196*512). The output of VGG-19 is further passed to a dense layer that has 250 hidden

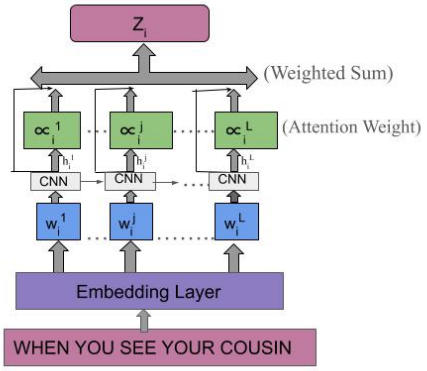


Figure 3: Attention for text

neurons. The output of this dense layer is passed to the attention layer to find out the important regions for classification.

3.2.1 Attention for Image

Image attention has been proven to be beneficial for many vision-related tasks (Zhou et al., 2019). We apply the attention over the image regions (output of dense layer) to find out the most important regions. Equation 6 shows the sequence of region maps for i_{th} image.

$$R_i = r_i^1, r_i^2, ...r_i^j, ...r_i^C \tag{6}$$

where, C is the number of regions and each region is now represented in D (250) dimension.

Attention score β_i^j is calculated for each region, signifying the region importance. It lies in the range between 0 and 1. If a region is more important for classification, then value of β_i^j will be more. Attention score β_i^j is calculated as shown in Equation 7

$$\beta_i^j = \frac{exp(p_i^j)}{\sum_{j=1}^{D} exp(p_i^j)} \tag{7}$$

Where,

$$p_i^j = \phi(Mr_i^j + b) \tag{8}$$

The weight matrix M and bias b are the parameters to be learned. ϕ is a nonlinear activation function and we use $tanh$ function. Finally, image features are calculated as weighted sum over all regions as shown in Equation 9.

$$U_i^k = \sum_{1<=j<=L} \beta_i^j r_i^j \tag{9}$$

The architecture of the attention for image is illustrated in Figure 4.

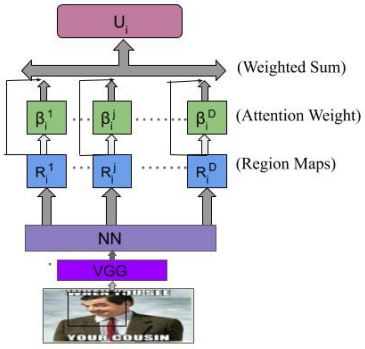


Figure 4: Attention for image

Type	Positive	Negative	Neutral	Total
Train	4155	629	2218	7002
Test	1109	173	593	1875

Table 1: SemEval dataset

3.2.2 Fusion of Text and Image features

Finally, the attended image features vector and text features vector are passed to a fully connected layer containing hidden neurons. This layer tries to find out the relation between image and text features and finally combines both.

3.3 Output Layer

The output of dense layer, i.e., the combined feature vector of image and text is finally passed to the output layer, which contains softmax as an activation function. The output layer maps the combined feature vector to a probability score. This probability score helps to classify the tweet into one of its predefined categories.

4 Dataset and Experiment

In this section, we discuss about the dataset used for the experiment, data collection, data annotation, and experimental details.

4.1 Dataset

We use the SemEval-2020 task 8 dataset[1] for sentiment analysis of memes. This dataset contains 8877 memes annotated for 3 classes, *viz.,* positive, negative, and neutral. The dataset is divided into 2 parts, training, and test. The distribution of the dataset is shown in Table 1.

As shown in Table 1, the data set is highly imbalanced. There are very less number of instances in

[1]https://competitions.codalab.org/competitions/20629

Dataset	Positive	Negative	Neutral
9277	4109	2375	2811

Table 2: Class-wise distribution

Type	Positive	Negative	Neutral
Train	2557	1907	1875
Development	443	295	343
Test	1109	173	593

Table 3: Data statistics

the negative class. So we crawl some data to make it balanced.

4.2 Data Collection and Annotation

We collect the memes from Reddit. After data collection, we extract the text written on memes using a python library known as python-tesseract. Python-tesseract is an optical character recognition (OCR) tool for python. After extracting text with Python-tesseract, we manually verify the output to correct the wrong instances. Then we conduct manual annotations for memes. Three annotators with post-graduate level knowledge in English are employed for annotations. Annotators are asked to write the overall polarity of the tweet for 3 classes, *viz.,* neutral, negative, and positive. Initially, to build an understanding of the class labels, we provide some tweets to the annotators with gold labels.

We added the newly annotated instances for negative class to the training part of the SemEval dataset. After merging, we divide it into two parts, train and validation. The test set is the same as provided in the original SemEval dataset. We downsample the positive class data for the balancing purpose. Class wise distribution of combined dataset is shown in Table 2. Train-dev-test distribution is shown in Table 3. [2]

4.3 Data Pre-processing

We perform the following steps to pre-process the text written memes.

- Convert all the characters of text into lowercase.
- Tokenize the sentence into sequence of words.
- Sentences with length less than maxlen are padded with zeros and greater than length maxlen are truncated.

4.4 Experimental Setup

We implement our model using python based Keras library [3]. We train our system for the 50 epochs and we save the checkpoints after every epoch to find the best performing model. We set the maximum sentence length to 80. We use batch size of 16 and ReLu activation function at the hidden layers of the network. We use optimizer Adam (Kingma and Ba, 2014) to optimize the weights of the network with a learning rate of 0.001. We use the softmax activation function at the last layer and categorical cross-entropy as the loss function. To prevent overfitting (Hawkins, 2004), dropout (Srivastava et al., 2014) of rate 0.5 is used at hidden layers. To find optimal values of hyper-parameters, we use the grid search.

4.5 Baseline models

We define the following baseline models.

- Baseline 1 (Textual model): Baseline 1 uses only textual information (text written on meme) for classification. We use the textual component without attention from the architecture shown in Figure 2.
- Baseline 2 (Visual model): Baseline 2 uses only visual information for classification. We use the image component without attention from the architecture shown in Figure 2.
- Baseline 3 (Textual model with attention): Baseline 3 uses only textual information and applies attention to the output of CNN to extract the most important words for classification as shown in Figure 2.
- Baseline 4 (Visual model with attention): Baseline 4 uses only visual information by extracting region features from VGG and apply attention over the regions to find out relevant regions for classification. Architecture is shown in Figure 2.
- Baseline 5 (Visual and textual without attention): Baseline 5 uses both textual as well as visual information for classification. We apply the architecture, as shown in Figure 2 by removing the attention layer from both image and text where image and textual features

[2] The annotated dataset is available from the authors upon request.

[3] https://keras.io/

Model	Macro F1 Score	Accuracy
Textual Model	31.42 %	43.25%
Visual Model	32.07%	42.16%
Textual Model With Attention	33.17%	44.34%
Visual Model With Attention	33.01%	42.98%
Visual And Textual Without Attention	33.22%	47.72%
SemEval Baseline	21.76	-
Proposed	34.23%	50.02%

Table 4: Evaluation results of different modalities

are concatenated and then passed to the fully connected layer.

- Baseline 6: Baseline 6 is provided by SemEval-2020 Task 8 which utilizes textual and image features.

- Final model: Our proposed model uses textual and visual information for classification by applying attention to text as well as image. Figure 2 describes our final architecture.

5 Evaluation Results

In this section, we discuss the detailed experimental results. We use accuracy and macro F1 score to evaluate the performance of our system. Table 4 shows the performance of our proposed model and comparison to the baseline models. The textual model (Baseline 1) yields the macro F1 and accuracy of 31.42% and 43.25%, respectively. Visual model (baseline 2) yields the macro F1 and accuracy of 32.07% and 42.16%, respectively. The model using only textual features with attention component (baseline 3) yields the macro F1 and accuracy of 33.17% and 44.34%, respectively. The visual model with attention (i.e., Baseline 4) yields macro F1 and accuracy of 33.01% and 42.98% ,respectively. Concatenation of textual and visual features (Baseline 5) without applying attention to image and textual features yields the macro F1 and accuracy of 33.22% and 47.72%, respectively. Reported macro F1 of SemEval baseline (Baseline 6) is 21.76%. Our proposed model obtains the macro F1 and accuracy of 34.23% and 50.02%, respectively. Our proposed system outperforms the other baselines, which indicates that multi-modal information actually helps to improve the effectiveness of the system. All the reported results are statistically significant as we have performed pairwise Welch's t-test (Welch, 1947) at 5% significant level.

Class	Negative	Neutral	Positive
Negative	17	43	113
Neutral	63	139	391
Positive	116	211	782

Table 5: Confusion matrix

5.1 Error Analysis

In this section, we present a detailed error analysis. Table 6 shows the example cases to establish the need for image as well textual model for sentiment classification of memes.

Column name is same as image name.

- Columns *a* shows the case where the textual model (Baseline 1) performs misclassification, but the visual model (Baseline 2) correctly predict the class.

- Column *b* shows the case where the visual model (Baseline 2) performs misclassification, but the visual model with attention model (Baseline 4) predict it correctly.

- Column *c* describes the case where the visual model (Baseline 2) is wrong, but the textual model (Baseline 1) performs correct classification.

- Column *d* shows the case where the textual model (Baseline 1) performs misclassification, but the textual model with attention (Baseline 3) performs correct classification.

- Column *e* shows the case when all the above-mentioned models perform misclassification, but the model which combine image and text through dense layer (Baseline 5) performs correct classification.

- Column *f* shows the case where all the baseline models fail, but our proposed model performs correct classification.

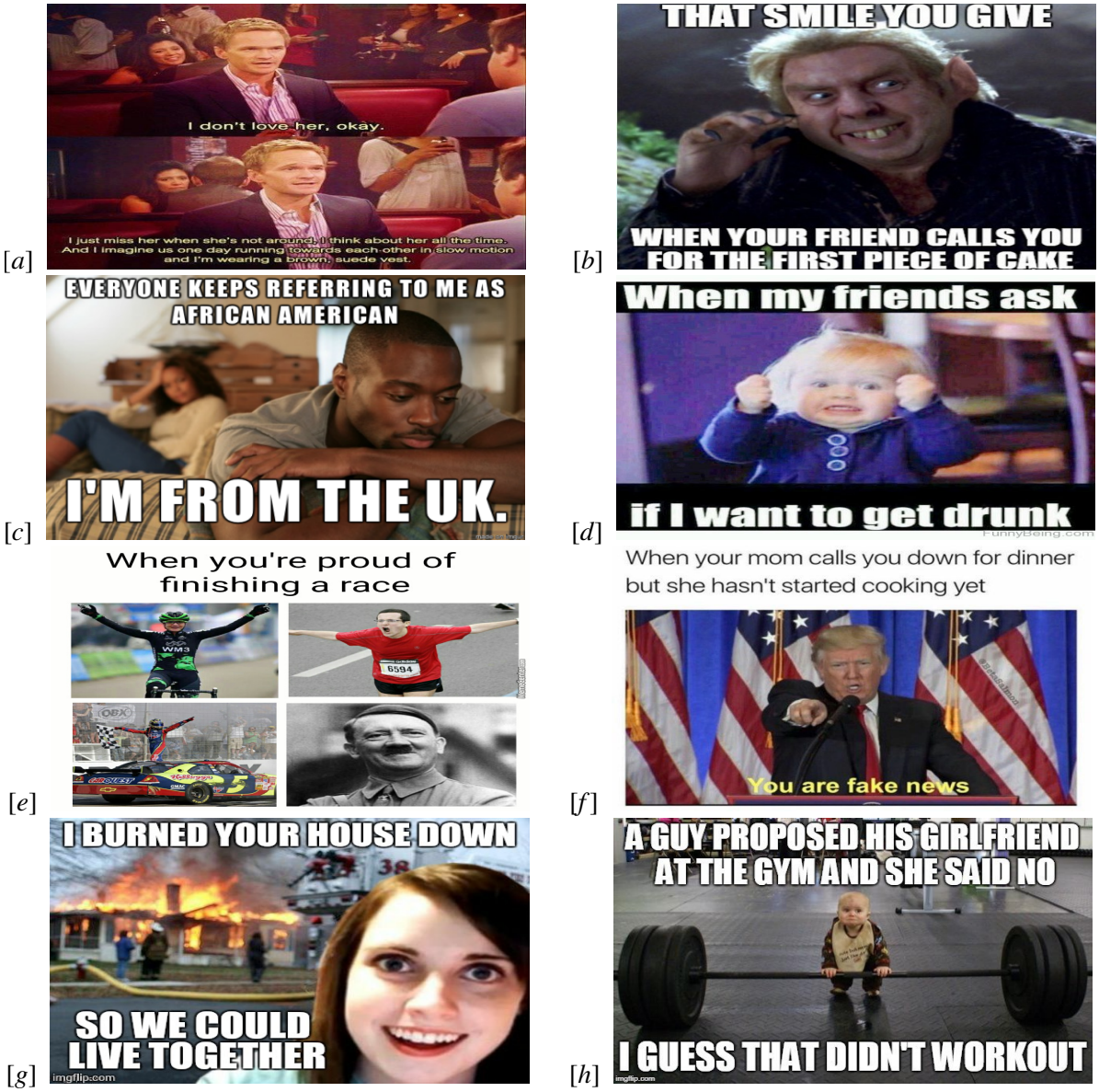


Figure 5: Qualitative analysis

Model	a	b	c	d	e	f
Actual	Neutral	Positive	Positive	Neutral	Neutral	Positive
Baseline 1	Negative	Negative	Positive	Positive	Positive	Negative
Baseline 2	Neutral	Neutral	Neutral	Positive	Positive	Negative
Baseline 3	Negative	Negative	positive	Neutral	Positive	Negative
Baseline 4	Neutral	Positive	Neutral	Positive	Negative	Neutral
Baseline 5	Negative	Neutral	Neutral	Positive	Neutral	Neutral
Proposed	Neutral	Positive	Positive	Neutral	Neutral	Positive

Table 6: Predictions of different models

Model	g	h
Actual	Negative	Neutral
Proposed Methodology	Neutral	Negative

Table 7: Qualitative error analysis of proposed model

These cases establish the effectiveness of our proposed approach. Further, we analyzed the output of our proposed model, both quantitatively and qualitatively. Confusion matrix is shown in the Table 5. It shows that the majority of negative class memes and neutral class got confused with positive class, and the majority of positive class memes got confused with neutral class. In Table 7, we show the cases where our proposed model performs misclassification. The column name is same as the image name. For image g, the proposed model misclassifies it to the neutral class, but the actual label is negative. A possible reason could be the presence of a happy face in the image. For image h, the predicted sentiment is negative, but the actual label is neutral. A possible reason could be the presence of a sad face in the image.

6 Conclusion

In this paper, we have proposed a multi-modal framework for meme sentiment classification by utilizing textual and visual information of memes. We use the SemEval-2020 task data and also annotated our own dataset to make this dataset balanced. We found that only textual information or only visual information is not sufficient to analyze a meme's sentiment. Our proposed framework utilizes textual and visual features and finally fuses both the information through a fully connected layer. Our proposed framework achieved the macro F1 and accuracy of 34.23% and 50.02%, respectively. Our proposed framework increases the accuracy by 6.77% and 7.86% compared to only textual and visual features, respectively. In the future, we are planning to explore other fusion methods to incorporate textual and visual features. We would also explore contextual embeddings for the text part of meme classifications.

References

Mamta ., Asif Ekbal, Pushpak Bhattacharyya, Shikha Srivastava, Alka Kumar, and Tista Saha. 2020. Multi-domain tweet corpora for sentiment analysis: Resource creation and evaluation. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5046–5054, Marseille, France. European Language Resources Association.

Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca J Passonneau. 2011. Sentiment analysis of twitter data. In *Proceedings of the workshop on language in social media (LSM 2011)*, pages 30–38.

Md Shad Akhtar, Asif Ekbal, and Erik Cambria. 2020. How intense are you? predicting intensities of emotions and sentiments using stacked ensemble. *IEEE Computational Intelligence Magazine*, 15(1):64–75.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Akshat Bakliwal, Jennifer Foster, Jennifer van der Puil, Ron O'Brien, Lamia Tounsi, and Mark Hughes. 2013. Sentiment analysis of political tweets: Towards an accurate classifier. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:P10008.

Hongshen Chen, Yue Zhang, and Qun Liu. 2016. Neural network for heterogeneous annotations. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 731–741.

Very Large Corpora. 2000. Empirical methods in natural language processing.

Deepak Chowdary Edara, Lakshmi Prasanna Vanukuri, Venkatramaphanikumar Sistla, and Venkata Krishna Kishore Kolli. 2019. Sentiment analysis and text categorization of cancer medical records with lstm. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–17.

Manoochehr Ghiassi and S Lee. 2018. A domain transferable lexicon set for twitter sentiment analysis using a supervised machine learning approach. *Expert Systems with Applications*, 106:197–216.

Douglas M Hawkins. 2004. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12.

Tao Jiang, Jiahai Wang, Zhiyue Liu, and Yingbiao Ling. 2020. Fusion-extraction network for multimodal sentiment analysis. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 785–797. Springer.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Akshi Kumar and Arunima Jaiswal. 2017. Image sentiment analysis using convolutional neural network. In *International Conference on Intelligent Systems Design and Applications*, pages 464–473. Springer.

Wenkuan Li, Peiyu Liu, Qiuyue Zhang, and Wenfeng Liu. 2019. An improved approach for text sentiment classification based on a deep neural network via a sentiment attention mechanism. *Future Internet*, 11(4):96.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

A Man, Yuanyuan Pu, Dan Xu, Wenhua Qian, Zhengpeng Zhao, and Qiuxia Yang. 2019. Multi-feature fusion for multimodal attentive sentiment analysis. In *MMAsia*, pages 43–1.

Walaa Medhat, Ahmed Hassan, and Hoda Korashy. 2014. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, 5(4):1093–1113.

Igor Mozetič, Miha Grčar, and Jasmina Smailović. 2016. Multilingual twitter sentiment classification: The role of human annotators. *PloS one*, 11(5):e0155036.

Gorti Satyanarayana Murty and Shanmukha Rao Allu. Text based sentiment analysis using lstm.

Alessandro Ortis, Giovanni Maria Farinella, Giovanni Torrisi, and Sebastiano Battiato. 2020. Exploiting objective text description of images for visual sentiment analysis. *Multimedia Tools and Applications*, pages 1–24.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Soujanya Poria, Navonil Majumder, Devamanyu Hazarika, Erik Cambria, Alexander Gelbukh, and Amir Hussain. 2018. Multimodal sentiment analysis: Addressing key issues and setting up the baselines. *IEEE Intelligent Systems*, 33(6):17–25.

Anthony Rios and Ramakanth Kavuluru. 2015. Convolutional neural networks for biomedical text classification: application in indexing biomedical articles. In *Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics*, pages 258–267.

Chhavi Sharma, Deepesh Bhageria, William Paka, Scott, Srinivas P Y K L, Amitava Das, Tanmoy Chakraborty, Viswanath Pulabaigari, and Björn Gambäck. 2020. SemEval-2020 Task 8: Memotion Analysis-The Visuo-Lingual Metaphor! In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2020)*, Barcelona, Spain. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Bernard L Welch. 1947. The generalization ofstudent's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35.

Guixian Xu, Yueting Meng, Xiaoyu Qiu, Ziheng Yu, and Xu Wu. 2019. Sentiment analysis of comment texts based on bilstm. *Ieee Access*, 7:51522–51532.

Quanzeng You, Liangliang Cao, Hailin Jin, and Jiebo Luo. 2016. Robust visual-textual sentiment analysis: When attention meets tree-structured recursive neural networks. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 1008–1017.

Jinfei Zhou, Yaping Zhu, and Hong Pan. 2019. Image caption based on visual attention mechanism. In *Proceedings of the 2019 International Conference on Image, Video and Signal Processing*, pages 28–32.

Towards Bengali Word Embedding: Corpus Creation, Intrinsic and Extrinsic Evaluations

Md. Rajib Hossain and Mohammed Moshiul Hoque
Department of Computer Science & Engineering, Chittagong University
of Engineering and Technology, Chittagong-4349 Bangladesh
rajcsecuet@gmail.com, moshiul_240@cuet.ac.bd

Abstract

Distributional word vector representation or word embedding has become an essential ingredient in many natural language processing (NLP) tasks such as machine translation, document classification, information retrieval and question answering. Investigation of embedding model helps to reduce the feature space and improves textual semantic as well as syntactic relations. This paper presents three embedding techniques (such as Word2Vec, GloVe, and FastText) with different hyperparameters implemented on a Bengali corpus consists of 180 million words. The performance of the embedding techniques is evaluated with extrinsic and intrinsic ways. Extrinsic performance evaluated by text classification, which achieved a maximum of 96.48% accuracy. Intrinsic performance evaluated by word similarity (e.g., semantic, syntactic and relatedness) and analogy tasks. The maximum Pearson ($\hat{r}$) correlation accuracy of 60.66% ($S_{s\hat{r}}$) achieved for semantic similarities and 71.64% ($S_{y\hat{r}}$) for syntactic similarities whereas the relatedness obtained 79.80% ($R_{s\hat{r}}$). The semantic word analogy tasks achieved 44.00% of accuracy while syntactic word analogy tasks obtained 36.00%.

1 Introduction

Word embedding is a distributional vector representation of words in which syntactic and semantic interpretations are derived from the enormous amount of unlabeled texts. Recently, the word embedding is considered as a powerful tool due to its many applications in NLP, thus, gained much attention by NLP experts. This is a growing up research issue for well-resourced language like English, where an embedding algorithm generates a model (Devlin et al., 2019). However, it is a very complicated task to adopt an embedding algorithm (of any language) directly for the resource-constrained languages such as Bengali due to the scarcity of resources. As a result, the low-resource language is trail end in NLP tools development. Bengali is the most popularly communicated language of Bangladesh while second most communicated of the 22 official languages of India which makes Bengali is the 7^{th} most spoken language in the world (Hossain and Hoque, 2019). However, due to the shortage of resources, the development of NLP tools are striving. Bengali speaking people are suffering to access modern NLP tools that might be affected their sustainable use of language technologies. Therefore, Bengali word embedding is an essential prerequisite to developing any Bengali language based NLP tools.

There are two well-known evaluation methods used extensively to evaluate embedding techniques such as extrinsic and intrinsic (Zhelezniak et al., 2019a). Extrinsic evaluation refers to downstream tasks like as machine translation (MT) (Banik et al., 2020), and Part of speech (POS) tagging (Priyadarshi and Saha, 2020). Intrinsic evaluation goals to evaluate the quality of language processing tasks, such as semantic and syntactic word similarity (Pawar and Mago, 2018), Word relatedness (Gladkova and Drozd, 2016), and Word analogy (Schluter, 2018). Unavailability of standard Bengali embedding corpus and inadequacy of resources are antecedents that make such a model generation and evaluation very challenging. Moreover, there is no generalized embedding model available to date for Bengali downstream tasks. Thus, the proposed work introduces a Bengali embedding model generation and evaluation techniques with different hyperparameters settings. Specifically, the key contributions of this work are:

- Acquire raw monolingual Bengali corpora with 180 million words where the unique words are 13 million.
- Construct and annotates the intrinsic and extrinsic evaluation datasets as well as evaluate the annotation purity.

Proceedings of the 17th International Conference on Natural Language Processing, pages 453–459
Patna, India, December 18 - 21, 2020.

- Generate *ninety* embedding models with the combination of three different algorithms (such as Word2Vec, GloVe,and FastText) and variations of model parameters.

- Examine the influence of hyperparameters on the embedding models performance.

As far as we are aware, the proposed work is the first attempt to generate large-scale embedding models evaluates with intrinsic and extrinsic evaluators.

2 Related Work

Distributional word vector representation or embedding model generation is a well-established research agenda in NLP domain. There are plenty of research works have been carried out on word embedding in high-resource languages, but it remains as a barrier for low resource languages. The first intrinsic evaluation datasets introduced in RG-65 (Rubenstein and Goodenough, 1965) which contains 65 contextual synonymy pairs. The WordSimilarity-353 introduced by (Finkelstein et al., 2001) and the dataset contains 353 words pair with 13 different subjects. In recent time, three embedding model evaluation datasets have been introduced: SimLex-999 (Hill et al., 2015), SemEval 2017 (Camacho-Collados et al., 2017), and MEN (Bruni et al., 2014). An Italian language embedding model has developed in (Di Gennaro et al., 2020), which achieved 53.74% overall analogies accuracy for 19,791 texts. Moreover, this work achieved a semantic analogies accuracy of 59.20% for 8,915 texts and syntactic accuracy of 48.80% for 10,876 texts. However, this work considered only 3COSADD similarity score and not consider the word relatedness and extrinsic evaluations. Ercan and Yıldız (2018) devised a Turkish word similarity and relatedness system that produced Turkish embedding dataset derived from English word similarity and relatedness datasets (e. g. WordSimilarity-353, MEN, and SimLex-999). This work achieved spearman score (ρ) of 0.667 for WordSimilarity-353, 0.68 for MEN and 0.67 for SimLex-999 respectively. The developed embedding model was not evaluated with extrinsic evaluation (Chiu et al., 2016).

Although most of the current works on embedding model, resource creation and evaluations conducted for the high-resource languages (e.g., English, Germany, and French) there are few comprehensive research conducted on low resource languages such as Assamese, Gujarati, Hindi, Kannada, and so on (Kumar et al., 2020) and Turkish (Ercan and Yıldız, 2018). Kumar et al. (2020) introduced the pre-trained word embedding models for Indian languages. The proposed system generates 14 Indian local language embeddings with 8 different approaches, including 436 models. Embedding models are evaluated by extrinsic evaluators and archived more than 90.00% accuracy for UPOS, and XPOS tagging using universal dependency treebank datasets (Nivre et al., 2016). The NER tagging accuracy about 95.00% with FastText embedding. Kumar et al. (2020) aimed to solve 14 Indian languages NLP problems, but the generated models are not considered intrinsic evaluations. To best of our knowledge, only single research was conducted concerning Bengali word embedding using Word2Vec (Sadman et al., 2019). However, this work considered only intrinsic evaluations with a self-build dataset. Our approach considered *ninety* embedding models based on GloVe, FastText and Word2Vec models and measured the performance using intrinsic and extrinsic evaluators.

3 Methodology

The principal aim of our research is to investigate the affect of intrinsic and extrinsic evaluations on Bengali word embedding models. Thus, the proposed scheme comprises of three main parts: corpus creation, word embedding model development, and evaluation. Figure 1 illustrates the abstract view of our work.

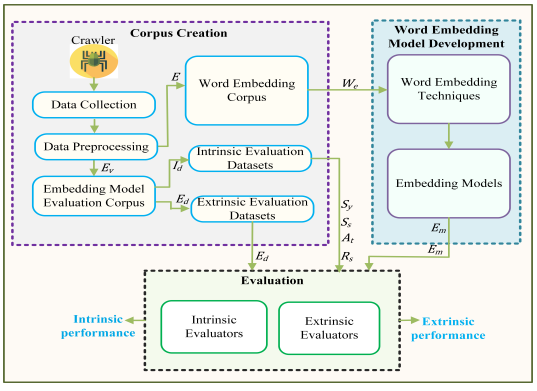


Figure 1: Abstract process of evaluations for Bengali word embedding.

3.1 Corpus Creation

We collected Bengali texts from various online sources and distributed these into two sets: word

embedding corpus (E) and embedding model evaluation corpus (E_v). We used a Python crawler to crawl data. We collected $910,720$ Bengali text files over a twenty-four month period (September 10, 2018, to September 11, 2020) which are forwarded to data preprocessing step. Initially, the non-Bengali alphabets and digits are removed from the text files. In the next, preprocessing step removed HTML tags, hashtags, URLs, punctuation and white spaces. Finally, the duplicate texts are deleted from the archive. The preprocessing step produced of $882,352$ usable text, and removed $28,368$ blank size text documents from the initial dataset due to various preprocessing operations. These usable preprocessed data are randomly distributed into two sets; one set for embedding model evaluation ($100,000$ texts) and another set for word embedding corpus ($782,352$ texts). The embedding corpus (W_e) (i.e., total $180,081,093$ words) is fed to the embedding techniques.

Embedding model evaluation corpus (E_v) is a combination of intrinsic (I_d) and extrinsic datasets (E_d). In order to perform an extrinsic evaluation, out of $100,000$ text documents, a total of $60,000$ documents are randomly selected. This dataset was labelled manually followed by majority voting to assign the suitable label. Two linguistic experts are assigned to annotated each data into one of the six pre-defined categories such as Accident (A_t), Crime (C_e), Entertainment (E_t), Health (H_h), Politics (P_s) and, Sports (S_p). Among $60,000$ text documents, both experts have been agreed upon $54,858$ text labels. The developed corpus (E_d) achieved a Kappa score (K) is 78.53%, which indicates a reasonable agreement between annotators for downstream task. To perform intrinsic evaluations of the embedding models four different sub datasets are used for conducting four measures: semantic word similarity (S_s), syntactic word similarity (S_y), relatedness (R_r), and analogy tasks (A_t). Intrinsic datasets and corresponding kappa score are shown in Table 1. Here, S_s, S_y and R_s datasets are substantial agreement where as A_t is moderate agreement.

Datasets	No. of samples	Kappa score (%)
S_s	100	71.55
S_y	100	65.30
R_s	100	75.13
A_t	100	56.90

Table 1: Embedding model datasets and kappa score.

3.2 Word Embedding Model Development

We consider three well-known embedding techniques for Bengali corpus including Word2Vec, GloVe, and FastText. To realize the effect of hyperparameters, we have considered embedding dimension ($size$), minimum word frequency count (min_count), contextual windows size ($window$) and number of iteration ($epoch$) for each of the embedding technique.

Word2Vec (skip-gram and CBOW): Word2Vec (Mikolov et al., 2013a) technique takes W_e as the input and produce the embedding model E_m using gensim library (Řehůřek and Sojka, 2010). Two versions of Word2Vec pre-trained models such as Skip-gram and continuous bag of words (CBOW) are used with similar hyperparameters for tuning: $size$: $\{50, 100, 150, 200, 250, 300\}$, $window$: $\{5, 10, 15\}$, min_count : $\{2\}$ and $epochs$: 30 respectively. There are 36 embedding models (e.g., 18 for CBOW and 18 for skip-gram) generated for W_e using Word2Vec technique.

GloVe: GloVe (Pennington et al., 2014) technique generated 18 embedding models (E_m) for the embedding corpus W_e. Different hyperparameters are used to optimize the model such as $size$: $\{50, 100, 150, 200, 250, 300\}$, min_count : $\{2\}$, X_MAX : 95, $epochs$: 30 and $window$: $\{5, 10, 15\}$ respectively. The remaining hyperparameters remains the same as default settings. Finally, the eighteen embedding models (E_m) are evaluated by the evaluators.

FastText (skip-gram and CBOW): FastText embedding technique (Bojanowski et al., 2017), takes W_e as input and generates a embedding model (E_m) as output using gensim library (Řehůřek and Sojka, 2010). Different values of hyperparameters are used to achieve optimize performance such as $size$: $\{50, 100, 150, 200, 250, 300\}$, $window$: $\{5, 10, 15\}$, min_count : $\{2\}$ and $epochs$: 30 respectively. Our approach produced of 36 models (e.g., 18 for FastText-skip-gram and 18 for FastText-CBOW) that are evaluated using evaluators.

4 Results and Discussion

Intrinsic evaluations are performed for a total of *ninenty* (e.g., Word2Vec=36, GloVe=18 and Fast-

			Semantic (%)		Syntactic (%)		Relatedness (%)	
Model	size	window	$S_{s\hat{\rho}}$	$S_{s\hat{r}}$	$S_{y\hat{\rho}}$	$S_{y\hat{r}}$	$R_{s\hat{\rho}}$	$R_{y\hat{r}}$
GloVe	**300**	**15**	**60.02**	**60.66**	69.54	70.62	79.20	79.72
	250	**10**	56.33	57.77	**70.41**	70.66	**79.22**	**79.80**
	250	**5**	56.93	57.90	69.87	**71.64**	79.20	78.33
FastText (SG)	250	10	53.64	53.38	39.47	38.31	68.28	67.54
	150	10	56.78	55.18	37.78	36.03	66.63	65.84
Word2Vec (SG)	250	5	44.31	45.10	31.78	30.51	56.43	57.41
FastText (Grave et al., 2018)	300	-	49.41	49.78	5.93	-1.55	47.23	43.91

Table 2: Performance of embedding models concerning semantic, syntactic and relatedness word similarity.

Text=36) embedding models. Among these, the results of best *four* embedding models presented for extrinsic and intrinsic evaluators.

4.1 Intrinsic evaluation results

The word similarity (semantic and syntactic) score is calculated by Cosine similarity (C). The model performance can be calculated from the spearman ($\hat{\rho}$) and pearson ($\hat{r}$) correlations (Zhelezniak et al., 2019b). The well-known word analogy solver, 3COSADD (Mikolov et al., 2013b) is used to solve the analogy tasks. Three similarity measures are used to evaluate word similarity and analogy tasks analysis. In order to maintain consistency, we performed training for all models with our developed corpus.

Word similarity: Table 2 shows the intrinsic evaluations performance of the embedding models. Annotators word similarity rates are range from $0-10$ whereas the cosine similarity score normalized by ten times. All values in Table 2 are normalized by hundreds times. Maximum semantic correlation values are $S_{s\hat{\rho}} = 60.02\%$ and $S_{s\hat{r}} = 60.66\%$ for GloVe ($size = 300$ and $window = 15$) technique. Highest syntactic correlation is $S_{y\hat{\rho}} = 70.41\%$ for GloVe ($size = 250$ and $window = 10$) where as $S_{y\hat{r}} = 71.64\%$ for GloVe ($size = 250$ and $window = 5$) techniques. The R_s highest correlations, $R_{s\hat{\rho}=79.22\%}$ and $R_{y\hat{r}=79.80\%}$ have been achieved using GloVe ($size = 250$ and $window = 10$) technique. There are *eight* semantic words pair are not able to process by E_m where as *four* syntactic words pair are not able to process by E_m of all techniques. Relatedness words pair are fully processed by all embedding techniques.

Word analogy results: The semantic analogy results are shown in Table 3, while Table 4 denotes the syntactic analogy tasks performance based on our corpus. Due to unavailability of Bengali semantic and syntactic analogy datasets, we have been developed A_t datasets were 50 analogy words used for semantic and another fifty used to perform syntactic analogy tasks. GloVe ($size = 300$ and $window = 15$) technique has achieved maximum accuracy of 38.00% (Add) and 44.00% (Mull) for semantic analogy tasks. Minimum semantic analogy tasks accuracy is obtained by FastText (Grave et al., 2018) embedding model. The maximum

Semantic analogy tasks accuracy (%)				
Model	size	window	Add	Mull
GloVe	**300**	**15**	**38.00**	**44.00**
FastText (SG)	250	10	30.00	34.00
Word2Vec(SG)	250	5	26.00	30.00
FastText (Grave et al., 2018)	300	-	20.00	26.00

Table 3: Analogy tasks performance summary for semantic datasets.

syntactic analogy tasks accuracy are 30.00% (Add) and 36.00% achieved by GloVe ($size = 300$ and $window = 15$) E_m, while 20.00% (Add) and 24.00% (Mull) from FastText (Grave et al., 2018) embedding model.

4.2 Extrinsic evaluation results

The E_d is a Bengali text classification dataset which partitioned into the three sets: training $(39,079)$, validation $(6,000)$ and testing $(9,779)$. The text classifier model trained with a multi-kernel CNN architecture (Kim, 2014). The performance of the text classifier model assesses with extrinsic

Syntactic analogy tasks accuracy (%)				
Model	size	window	Add	Mull
GloVe	**300**	**15**	**30.00**	**36.00**
FastText (SG)	250	10	26.00	28.00
Word2Vec(SG)	250	5	20.00	26.00
FastText (Grave et al., 2018)	300	-	20.00	24.00

Table 4: Analogy tasks performance for syntactic datasets.

evaluators including accuracy (A), micro average F1-score, average precision (A_p), average recall (A_r) and confusion matrix (CM) (Wu et al., 2020). The evaluators evaluated the embedding models (E_m) downstream task (e.g., text classification) performance (in Tables 5 and 6). Table 5 shows the summary of text classification performance.

Models	size/window	F1-score(%)	A(%)
Word2Vec	250/10	93.43	93.87
GloVe	**200/10**	**96.03**	**96.48**
FastText	200/15	95.57	95.71

Table 5: Extrinsic evaluation for text classification.

The GloVe model achieved the highest accuracy of 96.48%. For clarity, we presented only the results of best four embedding models out of ninety models. Table 6 depicts the confusion matrix of GloVe model ($size = 200$ and $window = 10$) for text classification performance. The maximum correctly predicted class is *Politics* and incorrectly predicted class is *Crime*. The highest misclassification occurred for *Crime* and *Accident* pair.

CM	A_t	C_e	E_t	H_h	P_s	S_p
A_t	1636	43	1	3	3	2
C_e	62	1468	2	10	27	3
E_t	1	4	1603	12	8	16
H_h	1	5	16	1593	12	9
P_s	3	15	3	11	1570	6
S_p	1	6	43	4	12	1565

Table 6: Confusion matrix of text classification task.

Figure 2 shows few example scores for semantic, syntactic and relatedness words pair score obtained from GloVe model and human annotators. GloVe and FastText (SG) models accuracy are considerable for semantic and relatedness similarities. In the case of extrinsic evaluations, the performance

	$Word_1$	$Word_2$	Annotators score (Avg.)	Cosine Similarity (C)
Semantic Words pair	অপরাধ Aparādha Crime	পাপ pāpa Sin	4.00	3.10
Semantic Words pair	জল Jala Water	পানি Pāni Water	8.80	6.69
Relatedness Words pair	স্কুল Skula School	কলেজ kalēja College	8.50	7.62
Relatedness Words pair	শার্ট Śārṭa Shirt	প্যান্ট Pyānṭa Pants	8.70	8.32
Syntactic Words pair	শিক্ষক Śikṣaka Teacher	শিক্ষকরা Śikṣakarā Pants	7.3	6.60
Syntactic Words pair	পর্বত Parbata Mountain	পর্বতমালা Parbatamālā Mountains	4.30	3.25

Figure 2: Word pair similarity scores, the Cosine similarity score is normalized by 10 times and annotators score is ranging between 1 to 10.

of GloVe and FastText embedding models are significant for the text classification task.

5 Conclusion and Future Work

In this work, we have been generated about *ninety* embedding models for the Bengali language. These models have developed using the combinations of three embedding techniques (such as GloVe, Word2Vec, and FastText) and various hyperparameters. All models have evaluated by extrinsic and intrinsic evaluators on our developed corpus. The performance of an embedding model significantly depends on the hyperparameters, corpus and nature of the model. Although GloVe model performed better than Word2Vec and FastText, there is no generalized embedding model for intrinsic and extrinsic NLP tasks. The embedding models are highly corpus oriented, and hyperparameters also vary from one task to another. In the future, the existing Bengali corpus can be extended for embedding model generation to alleviate the out-of-vocabularies problems. The context-dependent feature represents technique (such as BERT, ElMo and XLNet) will be investigated to find suitable embedding technique for Bengali. In addition to that, more analogy tasks can be considered to assess the performance of different embedding models with various intrinsic and extrinsic evaluators.

References

Debajyoty Banik, Asif Ekbal, and Pushpak Bhattacharyya. 2020. Statistical machine translation based on weighted syntax–semantics. *Sādhanā*, 45:1–12.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and

Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49(1):1–47.

Jose Camacho-Collados, Mohammad Taher Pilehvar, Nigel Collier, and Roberto Navigli. 2017. SemEval-2017 task 2: Multilingual and cross-lingual semantic word similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 15–26, Vancouver, Canada. Association for Computational Linguistics.

Billy Chiu, Anna Korhonen, and Sampo Pyysalo. 2016. Intrinsic evaluation of word vectors fails to predict extrinsic performance. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 1–6, Berlin, Germany. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Giovanni Di Gennaro, Amedeo Buonanno, Antonio Di Girolamo, Armando Ospedale, Francesco A. N. Palmieri, and Gianfranco Fedele. 2020. An analysis of word2vec for the italian language. *Smart Innovation, Systems and Technologies*, pages 137–146.

Gökhan Ercan and Olcay Taner Yıldız. 2018. AnlamVer: Semantic model evaluation dataset for Turkish - word similarity and relatedness. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3819–3836, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on World Wide Web*, page 406–414, New York, NY, USA. Association for Computing Machinery.

Anna Gladkova and Aleksandr Drozd. 2016. Intrinsic evaluations of word embeddings: What can we do better? In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 36–42, Berlin, Germany. Association for Computational Linguistics.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, Miyazaki, Japan. European Languages Resources Association (ELRA).

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Md. Rajib Hossain and Mohammed Moshiul Hoque. 2019. Automatic bengali document categorization based on deep convolution nets. In *Emerging Research in Computing, Information, Communication and Applications*, pages 513–525, Singapore. Springer Singapore.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Saurav Kumar, Saunack Kumar, Diptesh Kanojia, and Pushpak Bhattacharyya. 2020. "a passage to India": Pre-trained word embeddings for Indian languages. In *Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL)*, pages 352–357, Marseille, France. European Language Resources association.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).

Atish Pawar and Vijay Mago. 2018. Calculating the similarity between words and sentences using a lexical database and corpus statistics. *CoRR*, abs/1802.05667.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference*

on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Ankur Priyadarshi and Sujan Kumar Saha. 2020. Towards the first maithili part of speech tagger: Resource creation and system development. *Computer Speech Language*, 62:101054.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Commun. ACM*, 8(10):627–633.

Nafiz Sadman, Akib Sadmanee, Md. Iftekhar Tanveer, Md. Ashraful Amin, and Amin Ahsan Ali. 2019. Intrinsic evaluation of bangla word embeddings. In *2019 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pages 1–5. IEEE.

Natalie Schluter. 2018. The word analogy testing caveat. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 242–246, New Orleans, Louisiana. Association for Computational Linguistics.

Di Wu, Mengtian Zhang, Chao Shen, Zhuyun Huang, and Mingxing Gu. 2020. Btm and glove similarity linear fusion-based short text clustering algorithm for microblog hot topic discovery. volume 8, pages 32215–32225. IEEE Access.

Vitalii Zhelezniak, Aleksandar Savkov, April Shen, and Nils Hammerla. 2019a. Correlation coefficients and semantic textual similarity. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 951–962, Minneapolis, Minnesota. Association for Computational Linguistics.

Vitalii Zhelezniak, Aleksandar Savkov, April Shen, and Nils Hammerla. 2019b. Correlation coefficients and semantic textual similarity. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 951–962, Minneapolis, Minnesota. Association for Computational Linguistics.

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of LREC 2010 workshop New Challenges for NLP Frameworks*, pages 46–50, Valletta, Malta. University of Malta.

Annotated Corpus of Tweets in English from Various Domains for Emotion Detection

Soumitra Ghosh[1], **Asif Ekbal**[1], **Pushpak Bhattacharyya**[1], **Sriparna Saha**[1],
Vipin Tyagi[2], **Alka Kumar**[2], **Shikha Srivastava**[2], **Nitish Kumar**[2]
[1]Indian Institute of Technology Patna, India
{1821cs05, asif, sriparna, pb}@iitp.ac.in
[2]Centre for Development of Telematics (C-DOT, India)
{vipin, alkakm, shikha, nitish}@cdot.in

Abstract

Emotion recognition is a very well-attended problem in Natural Language Processing (NLP). Most of the existing works on emotion recognition focus on the general domain and in some cases to specific domains like fairy tales, blogs, weather, Twitter etc. But emotion analysis systems in the domains of security, social issues, technology, politics, sports, etc. are very rare. In this paper, we create a benchmark setup for emotion recognition in these specialised domains. First, we construct a corpus of 18,921 tweets in English annotated with Paul Ekman's six basic emotions (*Anger, Disgust, Fear, Happiness, Sadness, Surprise*) and a non-emotive class *Others*. Thereafter, we propose a deep neural framework to perform emotion recognition in an end-to-end setting. We build various models based on Convolutional Neural Network (CNN), Bi-directional Long Short Term Memory (Bi-LSTM), Bi-directional Gated Recurrent Unit (Bi-GRU). We propose a Hierarchical Attention-based deep neural network for Emotion Detection (HAtED). We also develop multiple systems by considering different sets of emotion classes for each system and report the detailed comparative analysis of the results. Experiments show the hierarchical attention-based model achieves best results among the considered baselines with accuracy of 69%.

1 Introduction

Online social media provides a platform for people to share their perspectives on various issues with their close ones or in the public forum. Twitter is a very popular and heavily used platform among the most social media users. The USA and India are the 1st and 3rd leading countries based on number of twitter users as of July 2020[1]. Twitter data serves as a rich source for text analysis tasks as this type of data is not very long, yet very rich in emotion content. This type of communication is free from barriers of age, race, culture, gender, etc. In recent times, understanding people's opinion and sentiment have been the need of the hour to meet various purposes, such as real-time trending, better customer service, winning elections, etc. Particularly in Indian context, we often hear stories how a single social-media post has changed the life of an individual or an organization in a positive[2] or negative way.

Sentiment analysis or opinion mining deals with the automatic identification and extraction of the underlying subjective information from text. It is often synonymously described as 'polarity detection' which is concerned with classifying an instance of data as 'positive', 'negative' or 'neutral'. Contrary to sentiment analysis, the emotional analysis relies on a more fine-grained analysis of the subjectivity information. It deals with the deeper analysis of human emotions and sensitivities. Emotion analysis goes a step further into a person's motives and impulses. It gives valuable and exact insights that are easily transformed into actions. It is usually based on a wide spectrum of moods rather than a couple of static categories. Inside positive sentiment, it detects specific emotions like happiness, pride, satisfaction, thankfulness or excitement, depending on how it is configured. Similarly, negative sentiment may span a variety of emotions like anger, sadness, fear, hopelessness, blame, etc. Usually, sentiment or emotion analysis works better on subjective texts (texts having emotions or feelings) than objective ones (statements or facts).

In this work, we focus on building a corpus of emotion-annotated tweets from the relevant topics

[1]https://www.statista.com/statistics/242606/number-of-active-twitter-users-in-selected-countries/

[2]How social-media revived a helpless old couple running an eatery 'baba ka dhaba': https://www.instagram.com/p/CGDAHGxlGTv/

Proceedings of the 17th International Conference on Natural Language Processing, pages 460–469
Patna, India, December 18 - 21, 2020.

of interest of recent times (i.e. social issues, technology, cyber-security etc.) that are quite a buzz among today's online social platform users. Most of the available corpora for emotion are from the domains limited to blogs, weather, elections, fairy tales and some more. There are some available corpora in the general domain too, but they mostly cover tweets from politics, world news, sports, etc. At the present, hardly any corpora cover diverse domains. We consider Paul Ekman's (Ekman, 1992) basic emotions *(Anger, Disgust, Fear, Happiness, Sadness, Surprise)* for the emotion labelling task. A non-emotive label *Others* is also introduced for tweets which do not fall within the scope of Ekman's basic emotions (Ekman, 1992).

For effective usage of the dataset, we develop multiple single-task models for emotion classification. We develop CNN, Bi-GRU and Bi-LSTM based deep learning models for the emotion classification task. We propose a Bi-GRU based framework with hierarchical attention (Yang et al., 2016) mechanism to extract important information from each sentence in a tweet effectively. We also develop a couple of models on a sub-set of emotion classes (4 classes and 6 classes) and perform a comparative analysis with the developed systems with 7 classes. The proposed hierarchical attention system for 7 classes attains superior results than the considered baselines with an overall test accuracy of 69% for the emotion classification task.

The rest of the paper is organized as follows. In section 2, we discuss some of the existing work and corpora concerning emotion analysis. Various aspects of resource creation, challenges and analysis are discussed in Section 3. Next, we discuss the methodologies we implement in Section 4. In Section 5, we discuss the implementation details, results and qualitative error analysis. Finally, we conclude in Section 6 and acknowledge the funding agency for this work in Section 7.

2 Background

In the past decades, several annotated corpora have been created for emotion recognition from texts. Various annotation schemes were introduced to serve the specific purpose for which the corpus is created. (Scherer and Wallbott, 1994) collected questionnaires answered by people with different cultural backgrounds to form *The International Survey on Emotion Antecedents and Reactions (ISEAR)* dataset. People reported on their emotional events. The dataset contains a total of 7,665 sentences from reports by approximately 3,000 respondents. Sentences are annotated with single labels, chosen from the set of following labels: *joy, fear, anger, sadness, disgust, shame, and guilt*. The *Affective Text* task (Strapparava and Mihalcea, 2007) in SemEval 2007 was proposed to focus on the emotion classification of news headlines extracted from news web sites. Given a set of predefined six emotion labels (Paul Ekman's basic emotions (Ekman, 1992)), classify the titles with the appropriate emotion label and/or with a valence indication *(positive/negative)*. (Aman and Szpakowicz, 2007) published a dataset of blog content consisting of 5,205 sentences from 173 blogs. Each instance is annotated with an emotion label from Ekman's basic emotions (Ekman, 1992) and also with an intensity score for that emotion. (Alm, 2008) researched the text-based emotion prediction problem in the literature domain. The author provided an annotated corpus of 15,302 sentences from 176 stories annotated from among the following seven emotion classes *(angry, disgusted, fearful, happy, sad, positive surprise, and negative surprise)*.

Crowdflower's dataset, *The Emotion in Text*[3], is a noisy single-labelled crowd-sourced annotated corpus of tweets. It primarily follows Plutchik's 8 basic emotions (Plutchik, 2001) in addition to another 3 emotions *(love, confusion and no emotion)*. The *Electoral-Tweets* dataset, published by (Mohammad et al., 2015), targets the domain of elections (2012 US Presidential election). It consists of over 100,000 crowdsourced responses to two detailed online questionnaires (the questions targeted emotions, purpose, and style in electoral tweets). (Ghazi et al., 2015) published the Emotion-Stimulus dataset to predict the cause of emotion in the text. The dataset consists of 820 sentences which are annotated both with emotions (one label per sentence) and their causes, and 1,549 sentences which are marked only with their emotion. Ekman's basic emotions (Ekman, 1992) with an added class *Shame* have been used for the annotation. *The Hashtag Emotion Corpus*, also known as *Twitter Emotion Corpus (TEC)*, was published by (Mohammad and Kiritchenko, 2015), and consists of 21,051 tweets. This resource was created to understand if emotion-word hashtags can successfully be used as emotion labels. Ekman's basic emotions

[3] https://data.world/crowdflower/sentiment-analysis-in-text

(Ekman, 1992) have been considered for the annotation process. Tweets were scraped that contained hashtags in the form *#emotion* corresponding to Ekman's (Ekman, 1992) 6 basic emotions (like #anger, #disgust).

DailyDialogs is a dataset of dialogs published by (Li et al., 2017) spanning over a variety of topics and better structured than any social media data. The SSEC corpus (Schuff et al., 2017) is an annotation of the SemEval 2016 Twitter stance and sentiment corpus (Mohammad et al., 2017) with Plutchik's emotion labels (Plutchik, 2001). The authors studied the relation between emotion annotation and the other annotation layers like stance and sentiment. The *EmoInt* dataset published by (Mohammad and Bravo-Marquez, 2017) for evaluation of the *WASAA-2017 Shared Task of Emotion Intensity (EmoInt)* contains 7,097 tweets annotated with a pair of emotion tag and intensity score of the corresponding emotion. The annotation was done via crowdsourcing with primarily (but not limited to) one among the following 4 emotions *anger, joy, sadness, and fear* and their respective intensity score ranging between 0 to 1. The *Affect in Tweets Dataset* of the SemEval 2018 Task 1 (Mohammad et al., 2018) was introduced to determine the intensity of both emotion and sentiment as well as multi-label emotion classification of tweets.

Recent works have shown the effectiveness of multi-task systems by learning several correlated tasks simultaneously (Akhtar et al., 2019; Majumder et al., 2019; Akhtar et al., 2020).

3 Corpus Creation

In this section we briefly describe the data creation process starting from the collection, pre-processing, annotation and inter-annotator reliability.

3.1 Data Collection

We use the Twython [4] python library (wrapper) to extract tweets from Twitter's Standard search API [5]. Tweets were extracted using certain domain-specific keywords *(terror, cyber-security, technology)* and their combinations between the time interval January 2018 and August 2019. Several hundred thousand tweets were collected initially. This was filtered using various lexicons to increase the coverage of the affect oriented tweets. Irrelevant (questions, requests, poems) and code-mixed instances were removed. Before using the corpus for our experiments, we perform some basic pre-processing to remove noise from the data (removing URLs [6], mentions, non-ASCII characters, punctuations except '.', '!' and '?', conversion to lowercase, etc). Smileys are replaced by their meanings (for example: *:-(* as *sad*, *<3* as *Love*). Frequently used contractions are replaced by their full versions (for example: *can't* as *cannot*, *he's* as *he is*). Each tweet is sentence tokenized considering '.', '!' and '?' as the sentence delimiters.

3.2 Data Annotation

The entire dataset is manually annotated by three annotators with the single-labelling scheme at the document level, that is, a tweet can have at most one emotion label. Some pre-annotated instances from each category of emotion were prepared to be shared with the annotators to facilitate the annotation process. Here, we use Ekman's (Ekman, 1992) basic emotions that have been widely used among several emotion classification tasks so far. In addition to these 6 basic emotions *(Anger, Disgust, Fear, Joy, Sadness, Surprise)*, we introduce a non-emotive class *Others* to mark those instances that do not fall in the above emotion categories. A few annotation samples are shown in Table 1.

3.3 Inter-Annotator Agreement

We measure the agreement among different annotators using Cohen's Kappa Statistic (Cohen, 1960) for the emotion task. The average agreement attained over the entire dataset was 0.74 which indicates that the annotations are of fair quality. Highest individual agreement attained was for the class *Joy* (0.87) followed by *Others* (0.83). Lowest attained agreement score (0.62) was for the *Surprise* class. This may be attributed to two reasons: the very low number of instances in the surprise class and the presence of both positive and negative types of surprise instances.

3.4 Corpus Analysis

Looking into the content of the corpus, we observe that certain words frequently occur in instances across all the emotion classes. For understanding the role of a certain entity (or event) as a contributing factor towards generating a particular kind of emotion, we observe some frequently

[4] https://pypi.org/project/twython/

[5] https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets

[6] We use Beautifulsoup to remove URLs: https://pypi.org/project/beautifulsoup4/

Actual tweet	Emotion
Tropical Cyclone Mona to Hit Fiji Islands on Saturday January 5, 2019 https://t.co/nwknedBFba	Others
@congressdotgov Louis Farrakhan promotes terrorism and racism. Why doesn't anyone talk about him calling white people Satan?	Anger
@WillieHarveyJr @CycloneFB Thanks for being a CYCLONE!!! Honored to have you!	Joy
Crime will not go WAY DOWN because of a border wall, sir. There is crime from US CITIZENS EVERYDAY. https://t.co/HOg4EfEfnP	Sadness

Table 1: Samples of annotated tweets from our dataset

occurring words (like terrorism, technology, crime, etc.) and their frequencies of occurrences among all the emotion classes. It is found that although some words tend to occur across all the emotion classes, their frequencies of distribution differ considerably based on the type of word and the nature of the emotion. Words like *terrorism, weapons* occur more frequently in classes like *anger, disgust, fear* than in *joy, sadness, others* classes. The annotated dataset has a very highly skewed distribution of instances over the 7 classes that we consider. There are four severely under-represented classes, namely *(disgust, fear, sadness and surprise)* and one over-represented class *(others)*.

4 Methodologies

We develop various deep learning-based multi-task models for automatic detection of emotion and its intensity. As base learning techniques, we use Convolution Neural Network (CNN) (Kim, 2014), Long Short Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) network (Cho et al., 2014). We build three separate multi-task models (CNN based, Bi-GRU based and Bi-LSTM based) on top of pre-trained word embedding (GloVe [7] (Pennington et al., 2014)). The embedding layer is initialized with the pre-trained weights and is learned during the training in accordance with our dataset. We employ word attention (Bahdanau et al., 2014) mechanism to focus on the informative words in a document (tweet) and obtain an aggregated representation(document vector) which is passed through two fully-connected layers (100 neurons in each layer) and an output layer (with 7 neurons, one for each class) with Softmax activation. We use the categorical cross-entropy as the loss function.

[7] http://nlp.stanford.edu/data/wordvecs/glove.840B.300d.zip

4.1 Convolution Neural Network (CNN)

In the past few years, CNN has produced some break-through results in various NLP tasks. CNN relies heavily upon two operations for extracting features: convolution (produce feature maps) and pooling (dimension reduction). (Kim, 2014); (Akhtar et al., 2016); (Singhal and Bhattacharyya, 2016) have used CNN in different sentiment analysis tasks. Our CNN based classification system employs 3 convolution layers in parallel with 100 filters of sizes 2, 3 and 4 respectively. The output of the layers is added (merged) to produce a single output of the same shape as the individual layer's output. Max pooling operation (poolsize = 2) is performed on the convoluted output which is further passed through an attention layer (Bahdanau et al., 2014) to get an aggregated representation (document vector) of the informative words in a document (tweet). Lastly, the output from the attention layer is passed through two fully connected layers (with 100 neurons in each layer) with ReLu activation (Glorot et al., 2011) and an output layer (with 15 neurons, one for each class) with Softmax activation.

4.2 Bidirectional Long Short Term Memory Network (Bi-LSTM)

LSTMs (Hochreiter and Schmidhuber, 1997) are well known for their ability to preserve long term dependencies in the text, thus eliminating the vanishing gradient problem. LSTM employs 3 gates (forget, input and output) for regulating the amount

of information it wants to retain in its cell state (memory). Bi-LSTMs have shown promising results for several applications. Bi-LSTMs run the inputs in both forward and backward passes (positive time direction and negative time direction) generating two hidden states which, when combined, preserve information from both past and future. We use a Bi-LSTM layer having 256 neurons with Tanh recurrent-activation and 25% dropout and recurrent-dropout. The encoded representation from the Bi-LSTM layer is passed through the word attention (Bahdanau et al., 2014) layer which is further passed through 2 fully connected layers (with 100 neurons in each layer) with ReLu activation (Glorot et al., 2011) and an output layer (with 7 neurons, one for each class) with Softmax activation.

4.3 Bidirectional Gated Recurrent Unit (Bi-GRU)

Unlike LSTMs where 3 gates are involved, GRUs has 2 gates (update and reset gate) to control the amount of information it wants to retain, making it simpler and faster internally than LSTMs. Bidirectional GRUs takes into account the use of information from both the past time steps and future time steps to make decisions about the present state. Unlike LSTMs, GRUs (Cho et al., 2014) do not have any explicit cell state (memory) but still handle the vanishing gradient problem and learn long-term dependencies by the help of its gates mechanism. We use a Bidirectional GRU layer having 256 neurons. Word attention (Bahdanau et al., 2014) is applied on the encoded output (from the GRU layer) which is further passed through 2 fully connected layers (with 100 neurons in each layer) with ReLu activation (Glorot et al., 2011) and an output layer (with 7 neurons, one for each class) with Softmax activation.

4.4 Hierarchical Attention Based Deep Neural Framework for Emotion Detection (HAtED)

In recent works, Hierarchical attention (Bahdanau et al., 2014) based deep learning systems have gained popularity because of their good and consistent performance in various classification tasks when compared to the existing state-of-the-art techniques. In this work, we try to exploit the advantages of such an approach to improve upon our attention mechanism by focusing on words (at the sentence level) as well as sentences (at document level).

HAtED focuses on each sentence in a tweet individually resulting in sentence vectors which are further attended upon to produce a document vector. The intuition is to focus upon important words in a sentence as well as important sentences in a document (tweet) for a particular emotion. For encoding of the sentences, we leverage Bi-GRU (256 neurons) based word encoder. Without making major changes to the basic architecture of the hierarchical attention framework as in the original work (Yang et al., 2016), we tweaked the last few layers to solve our objective. We pass the document vector through a dense layer (100 neurons with ReLU activation) followed by an output layer (7 neurons with Softmax activation). We use categorical cross-entropy loss function for the classification task.

Besides HAtED, we also develop two separate Hierarchical Attention-based models considering various sets of emotion classes. They are as follows:

- **HAtED$^{4\text{-C}}$**: We develop HAtED$^{4\text{-C}}$ following the same architecture as HAtED but considering a sub-set of the 7 classes as considered in HAtED. We take motivation from the WASSA-2017 shared task (Mohammad and Bravo-Marquez, 2017) on Emotion Intensity and consider instances from the following 4 emotion classes in our dataset: *Anger, Fear, Joy, Sadness*. Leaving out two of the severely under-represented classes to build HAtED$^{4\text{-C}}$ helps us to get a better approximation of the effectiveness of our proposed approach as the negative impact of having unbalanced dataset is considerably reduced.

- **HAtED$^{6\text{-C}}$**: In this setting, we do not consider the Others class and train our model on Ekman's 6 Basic Emotion Classes (Ekman, 1992) which are as follows *Anger, Disgust, Fear, Joy, Sadness, Surprise*. Overall architecture is similar to that of HEtED with only change being the number of neurons in the output layer (6 neurons).

All the models described in section 4 are trained and tuned independently. Training of models is done through backpropagation using the Adam optimizer (Kingma and Ba, 2014). We employ 25% Dropout (Srivastava et al., 2014) in all the fully connected layers to prevent over-fitting.

Emotion	Train	Test	Val	Total
Anger	2475	688	275	3438
Disgust	865	241	97	1207
Fear	445	123	49	617
Joy	2911	809	323	4043
Sadness	647	180	72	899
Surprise	242	67	27	336
Others	6033	1677	671	8381

Table 3: Distribution of instances over respective emotion classes.

5 Experiments

5.1 Experimental Settings

Dataset: For our experiments, we split our curated dataset into three parts: train, validation and test sets in a 70:20:10 ratio, respectively. As mentioned earlier, the dataset is highly skewed with several under-represented classes (disgust, fear, sadness, surprise). The data distribution over the various emotion classes is shown in Table 3.

Implementation: We use the Python-based libraries Keras and Scikit-learn for the implementation. We use 300-dimension GloVe (Pennington et al., 2014) pre-trained embeddings to initialize the embedding layer in our models which are further learned during training on our data to obtain emotion-enriched word representations. First, we develop three basic deep-learning models (CNN, Bi-GRU and Bi-LSTM) which have been extensively used in various classification tasks on textual data. Considering these models as baselines, we build three hierarchical attention based Bi-GRU systems for the emotion classification task. Two of these three systems are built for 4 (HAtED$^{4\text{-C}}$) and 6 (HAtED$^{6\text{-C}}$) emotion classes, respectively. The third one (HAtED) is a hierarchical attention based emotion detection system for 7 classes.

Evaluation metrics: As our dataset is unbalanced, we consider the macro-average measure of precision (P), recall (R) and F1-scores as our evaluating metrics for the emotion detection task. In this case, we also compute the overall test accuracy.

5.2 Results and discussion

Table 2 and table 4 show the per-class precision, recall, F1-score and accuracy values for all the implemented models. Scores for classes with fewer instances (*Disgust, Fear, Sadness, Surprise*) are not at par with that of the better-represented classes with (*Anger, Joy, Others*). The 4-class model (HAtED$^{4\text{-C}}$) achieves better scores compared to its 6-class (HAtED$^{6\text{-C}}$) and 7-class (HAtED) variants for those classes. This may be attributed to the lower degree variance in data because of the smaller number of classes. The hierarchical attention-based systems outperformed the base learning systems (i.e. CNN, Bi-GRU and Bi-LSTM) in terms of the various metrics considered. Overall performance of the models on the test set is shown in Table 5. We compare and evaluate the performance of those systems which are built on **7-classes** *(i.e. CNN, Bi-GRU, Bi-LSTM and HAtED)*. HAtED outperforms the other models (for 7 classes) with a test accuracy of 69% and macro-average F1-score of 0.46. Performance of CNN is better than the HAtED model in terms of precision. Results of the HAtED$^{4\text{-C}}$ and

	CNN				Bi-LSTM				Bi-GRU			
Emotion	**P**	**R**	**F1**	**A**	**P**	**R**	**F1**	**A**	**P**	**R**	**F1**	**A**
Anger	0.55	0.69	0.61	0.69	0.45	0.75	0.59	0.75	0.56	0.66	0.61	0.66
Disgust	0.29	0.01	0.02	0.01	0.60	0.01	0.03	0.01	0.26	0.16	0.20	0.16
Fear	0.75	0.03	0.05	0.03	0.64	0.06	0.11	0.06	0.41	0.17	0.24	0.17
Joy	0.80	0.75	0.77	0.75	0.79	0.79	0.79	0.79	0.79	0.78	0.79	0.78
Sadness	0.27	0.02	0.03	0.02	0.00	0.00	0.00	0.00	0.31	0.11	0.16	0.11
Surprise	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Others	0.67	0.84	0.75	0.84	0.71	0.80	0.75	0.80	0.71	0.82	0.76	0.82

Table 2: Per-class Precision (P), Recall (R), F1-score (F1) and Accuracy (A) values for the CNN, Bi-LSTM and Bi-GRU models

	HAtED$^{4\text{-C}}$				HAtED$^{6\text{-C}}$				HAtED			
Emotion	**P**	**R**	**F1**	**A**	**P**	**R**	**F1**	**A**	**P**	**R**	**F1**	**A**
Anger	0.81	0.85	0.83	0.85	0.64	0.87	0.74	0.87	0.66	0.78	0.71	0.78
Disgust	-	-	-	-	0.42	0.15	0.22	0.15	0.35	0.28	0.31	0.28
Fear	0.44	0.38	0.41	0.38	0.57	0.39	0.46	0.39	0.38	0.15	0.22	0.15
Joy	0.88	0.91	0.90	0.91	0.89	0.89	0.89	0.89	0.78	0.81	0.79	0.81
Sadness	0.62	0.45	0.52	0.45	0.44	0.49	0.46	0.49	0.32	0.11	0.11	0.11
Surprise	-	-	-	-	0.25	0.01	0.03	0.01	0.00	0.00	0.00	0.00
Others	-	-	-	-	-	-	-	-	0.73	0.79	0.76	0.79

Table 4: Per-class Precision (P), Recall (R), F1-score (F1) and Accuracy (A) values for the Hierarchical attention based Bi-GRU systems. '-' indicates no score corresponding to a particular metric for a specific emotion class as that particular system do not consider that emotion class during training.

HAtED$^{6\text{-C}}$ models show that less variance in data (fewer number of classes) enhances the decisive power of the models for the classes it is built upon.

We observe from 5 that when we consider a fewer number of classes during training our system (HAtED$^{4\text{-C}}$), reducing the negative impact of severely under-represented classes (disgust, surprise), the scores for all the metrics improves remarkably when compared to HAtED. The scores from the HAtED$^{6\text{-C}}$ model shows that leaving out the *Others* class has also resulted in improving the scores of HAtED system. In other words, the introduction of *Others* class leads to an increase in misclassifications by the HAtED system. This effect is quite similar to the situation of having *Neutral* class in training for a Sentiment classification task which eventually leads to performance degradation of the sentiment classifier.

5.3 Qualitative Analysis

We perform a detailed qualitative analysis of the results from the models that we developed. Table 6 (correct classifications) and Table 7 (incorrect classifications) show some samples of predictions by the HAtED model. Indeed, implicit tweets (i.e. instances not having any explicit mention of affect information) are the major sources of errors. For example:

- **Tweet**: @Ru_NRD Truueee. They said they got involved to fight terrorism and ISIS but we all know why they are really involved.
  Actual: *Disgust*; **Predicted**: *Joy*.

Tweets having multiple emotions (say, anger as well as disgust) seems to hinder the models' overall performance.

Models	**P**	**R**	**F1**	**Acc.**
(Considering 7 classes)				
Baselines				
CNN	**0.47**	0.33	0.32	0.66
Bi-GRU	0.43	0.38	0.39	0.67
Bi-LSTM	0.46	0.34	0.32	0.67
Proposed				
HAtED	0.46	**0.46**	**0.46**	**0.69**
Considering 4 classes				
HAtED$^{4\text{-C}}$	0.68	0.64	0.66	0.81
Considering 6 classes)				
HAtED$^{6\text{-C}}$	0.53	0.46	0.46	0.71

Table 5: macro-average Precision, Recall, F1-score values and Accuracy scores on the test set are shown in the table. Values in bold signify the best attained scores for the respective metrics among the 7-class models.

- **Tweet**: @nimish4fk @RatanSharda55 @kushal_mehra @vivekagnihotri Why are you surprised? Congress is the power that created naxalism in india and used it to grab and retain power.
  Actual: *Disgust*; **Predicted**: *Anger*.

It is observed that certain instances from *joy* and *others* class are misclassified as belonging to some negative emotion class. This is primarily due to the presence of word(s), in such instances, which have mostly occurred in negative contexts in the overall dataset.

6 Conclusion

In this work, we have proposed a benchmark deep learning setup for emotion detection. The

Tweet	Act_Emo	Pred_Emo
'Criminals are evolving their social engineering tactics in an attempt to trick even the most savvy individuals Stay alert to the latest scam strategies to avoid becoming a victim'	Anger	Anger
'Cyclone Penny re-forms and could still about face toward Queensland coast'	Fear	Fear
'DrumFit is a fun way to blend technology and physical education at Bear Bytes Tech Expo'	Joy	Joy
'#Cyberattacks Skyrocketed in 2018. Are You Ready for 2019? Meet the premier #cyber industry at #ISDEF2019! https://t.co/KUghSb2foD'	Others	Others

Table 6: Samples of correct predictions from the HAtED model. Act_Emo and Pred_Emo means Actual Emotion and Predicted Emotion respectively.

Tweet	Act_Emo	Pred_Emo
'For those who are thinking Casteism doesn't exist and saying don't divide Hindus Wake up, u were already divided by the Varna system made by Upper caste'	Anger	Disgust
'When your college finally gets a MS cyber security program!!!'	Joy	Others
'Schools Volcano Explosion Experiment Goes Horrifically Wrong In India'	Fear	Disgust
'I still remember when Arsenal lost 4-0 to Chelsea on my birthday. Terrorism.'	Sadness	Anger

Table 7: Samples of incorrect predictions from the HAtED model.

corpus introduced in this work has been built from diverse domains of tweets carrying various emotions. We have built baseline models with CNN, Bi-GRU and Bi-LSTM. The performance of the Bi-GRU variant has improved significantly when we leveraged the effectiveness of hierarchical attention mechanism in HAtED. Comparison of results has demonstrated that the HAtED model outperforms the baselines by a fair margin (69% test accuracy on the emotion classification task) showing the efficacy of our approach.

Scarcity of instances in some emotion classes have resulted in low per-class performances for those classes showing the scope of improvement for our proposed system. We intend to extend the dataset with the goal to get a balanced distribution of instances over all the classes. We would also like to address the present problem in a multi-label multi-class setting since it was observed that a considerable amount of tweets have shown the presence of more than one emotion. With the intuition that sentiment may play a positive role in assisting the emotion classification task, we are eager to build a parallel multi-task system for automatic emotion (primary task) and sentiment detection (secondary task).

References

Md Shad Akhtar, Asif Ekbal, and Erik Cambria. 2020. How intense are you? predicting intensities of emotions and sentiments using stacked ensemble. *IEEE Computational Intelligence Magazine*, 15(1):64–75.

Md Shad Akhtar, Ayush Kumar, Asif Ekbal, and Pushpak Bhattacharyya. 2016. A hybrid deep learning architecture for sentiment analysis. In *Proceedings*

of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 482–493.

Shad Akhtar, Deepanway Ghosal, Asif Ekbal, Pushpak Bhattacharyya, and Sadao Kurohashi. 2019. All-in-one: Emotion, sentiment and intensity prediction using a multi-task ensemble framework. *IEEE Transactions on Affective Computing*.

Ebba Cecilia Ovesdotter Alm. 2008. *Affect in* text and speech*. Citeseer.

Saima Aman and Stan Szpakowicz. 2007. Identifying expressions of emotion in text. In *International Conference on Text, Speech and Dialogue*, pages 196–205. Springer.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Paul Ekman. 1992. An argument for basic emotions. *Cognition & emotion*, 6(3-4):169–200.

Diman Ghazi, Diana Inkpen, and Stan Szpakowicz. 2015. Detecting emotion stimuli in emotion-bearing sentences. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 152–165. Springer.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. Dailydialog: A manually labelled multi-turn dialogue dataset. *arXiv preprint arXiv:1710.03957*.

Navonil Majumder, Soujanya Poria, Haiyun Peng, Niyati Chhaya, Erik Cambria, and Alexander Gelbukh. 2019. Sentiment and sarcasm classification with multitask learning. *IEEE Intelligent Systems*, 34(3):38–43.

Saif M Mohammad and Felipe Bravo-Marquez. 2017. Wassa-2017 shared task on emotion intensity. *arXiv preprint arXiv:1708.03700*.

Saif M. Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 Task 1: Affect in tweets. In *Proceedings of International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, USA.

Saif M. Mohammad and Svetlana Kiritchenko. 2015. Using hashtags to capture fine emotion categories from tweets. *Computational Intelligence*, 31(2):301–326.

Saif M Mohammad, Parinaz Sobhani, and Svetlana Kiritchenko. 2017. Stance and sentiment in tweets. *ACM Transactions on Internet Technology (TOIT)*, 17(3):26.

Saif M Mohammad, Xiaodan Zhu, Svetlana Kiritchenko, and Joel Martin. 2015. Sentiment, emotion, purpose, and style in electoral tweets. *Information Processing & Management*, 51(4):480–499.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Robert Plutchik. 2001. The nature of emotions: Human emotions have deep evolutionary roots, a fact that may explain their complexity and provide tools for clinical practice. *American scientist*, 89(4):344–350.

Klaus R Scherer and Harald G Wallbott. 1994. Evidence for universality and cultural variation of differential emotion response patterning. *Journal of personality and social psychology*, 66(2):310.

Hendrik Schuff, Jeremy Barnes, Julian Mohme, Sebastian Padó, and Roman Klinger. 2017. Annotation, modelling and analysis of fine-grained emotions on a stance and sentiment detection corpus. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 13–23.

Prerana Singhal and Pushpak Bhattacharyya. 2016. Borrow a little from your rich cousin: Using embeddings and polarities of english words for multilingual sentiment classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3053–3062.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Carlo Strapparava and Rada Mihalcea. 2007. Semeval-2007 task 14: Affective text. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 70–74.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.

PhraseOut: A Code Mixed Data Augmentation Method for Multilingual Neural Machine Translation

Binu Jasim
IIIT Hyderabad
binujasim.t
@research.iiit.ac.in

Vinay P Namboodiri
IIT Kanpur
vinaypn
@cse.iitk.ac.in

C.V. Jawahar
IIIT Hyderabad
jawahar
@iiit.ac.in

Abstract

Data Augmentation methods for Neural Machine Translation (NMT) such as back-translation (BT) and self-training (ST) are quite popular. In a multilingual NMT system, simply copying monolingual source sentences to the target (Copying) is an effective data augmentation method. Back-translation augments parallel data by translating monolingual sentences in the target side to source language. In this work we propose to use a partial back-translation method in a multilingual setting. Instead of translating the entire monolingual target sentence back into the source language, we replace selected high confidence phrases only and keep the rest of the words in the target language itself. (We call this method PhraseOut). Our experiments on low resource multilingual translation models show that PhraseOut gives reasonable improvements over the existing data augmentation methods.

1 Introduction

Data augmentation methods are popular in the field of computer vision. For example, it is a common practice to obtain extra training data by flipping and cropping images. Consistency regularization refers to the idea that a model should output the same label to an augmented example as the original example which in turn encourages distributional smoothness in the model (Berthelot et al., 2019). Consistency regularization has been used to obtain state of the art results in automatic speech recognition (S. Park et al., 2019) by randomly striking out horizontal and vertical portions of speech spectrogram. But for textual data, which is discrete, consistency regularization techniques are not easily applicable since changing a single word could change the entire meaning of a sentence. This work proposes a data augmentation method for textual data which doesn't change its original meaning and encourage consistency regularization.

Some of the existing approaches to data augmentation in Neural Machine Translation (NMT) are based on word replacements such as word dropout (Sennrich et al., 2016a; Gal and Ghahramani, 2016). A recent approach, termed SwitchOut (Wang et al., 2018) claims that randomly replacing words in the source and target sentences by words uniformly sampled from the respective vocabularies can improve neural machine translation. There are also attempts to inject artificial noise in the clean data according to the distribution of types of actual noise to make NMT systems more robust (Vaibhav et al., 2019).

Another approach to data augmentation is to make use of monolingual data. The most popular example is back-translation (Sennrich et al., 2016b) where a reverse translation model is employed to translate large amounts of target monolingual data back into source language. This (noisy) source and original target pair is added as additional parallel data and is shown to be useful while attempting to obtain state of the art results in machine translation (Edunov et al., 2018).

Multilingual NMT whereby sharing a single model between several languages has become a standard paradigm in NMT including industrial applications such as Google Translate (Johnson et al., 2017). Multilingual NMT has several advantages over training individual translation models for each language pair including faster inference, enabling zero shot machine translation and superior performance (Aharoni et al., 2019). A multilingual NMT model is particularly beneficial when training to translate low resource languages (Neubig and Hu, 2018).

The main contribution of this work is to show that a phrase based data augmentation strategy consistently provides improvement especially in low-resource language settings. We show that this is also useful in code-mixed translation settings.

Proceedings of the 17th International Conference on Natural Language Processing, pages 470–474
Patna, India, December 18 - 21, 2020.

2 Existing Data Augmentation Methods for NMT

Let $(\mathcal{X}, \mathcal{Y})$ denote the available parallel corpus. Our task is to find augmented parallel sentences $(\hat{x}, \hat{y})$ from the same distribution where the parallel data is sampled from.

After acquiring synthetic parallel data, it is appended to the available parallel data. An encoder-decoder based NMT model is trained to maximize the usual MLE objective as follows:

$$\sum_{n=1}^{N} \sum_{t=1}^{T^{(n)}} \log p_\theta(y_t^{(n)} | \mathbf{x}^{(n)}, y_{<t}^{(n)}) \quad (1)$$

Below we discuss several existing data augmentation techniques to generate synthetic parallel data.

Copying Monolingual Data (Copying): The monolingual target sentences are copied as the source sentence as well to create a synthetic parallel corpus to train (Currey et al., 2017; Burlot and Yvon, 2018). This is shown to improve the target side language fluency as it could learn from large amounts of monolingual sentences. This method is suitable for a multi-lingual setting as the encoder has to deal with sentences in more than one language. Note that we can combine word dropout with Copying.

Word Dropout: Dropping words from the source as well as target sentences (with a probability, say 0.1) has shown to improve the performance of NMT systems (Sennrich et al., 2016a). This could help learn the sentence representation better as in a denoising autoencoder (Vincent et al., 2008).

SwitchOut: This method proposes to replace a word (with a fixed probability, similar to the dropout probability) with another word from the vocabulary, chosen uniformly (Wang et al., 2018). SwitchOut is applied to both the source and the target sentences. Note that SwitchOut doesn't make use of any additional monolingual data. It performs augmentations only on the parallel data.

The paper report high variance in performance, because of which they run experiments 7 times and report the median performance. The performance was shown to be better than word dropout. Yet we find that data augmentation by Copying is much better than SwitchOut. Hence if extra monolingual data is available, then SwitchOut is of no practical significance.

Back Translation (BT): Along with the original NMT system, maintain a translation system in the reverse direction. Then each monolingual sentence is translated into the source sentence. This noisy source and good target pair can be used as a synthetic training data (Sennrich et al., 2016b). This has been shown to improve machine translation significantly to achieve new state-of-the-art results (Edunov et al., 2018). Yet Back Translation is of limited benefit if the initial reverse model is of poor quality (Wang et al., 2018).

Self Training (ST) Forward translate monolingual data using the same model to obtain target sentence. This source and noisy target pair can be used as synthetic augmented data along with parallel data (He et al., 2020). Self training gives similar performance improvements as BT.

In this work we don't compare against ST since it uses monolingual data in the source side while BT as well as other methods use monolingual data in the target side.

3 PhraseOut

We propose to use a partial back-translation technique in a multi-lingual setting. We do the augmentation only in the source side and keep the target sentence as it is similar to back-translation. The augmented sentence $\hat{y}$ is obtained by replacing a randomly chosen phrase from the target sentence y with a source language phrase. Hence we hope that both y and $\hat{y}$ are close in the semantic space and hence the distributional smoothness is maintained. This augmented sentence is copied as the source sentence.

PhraseOut is described in Algorithm 1. Using the available parallel data, we learn a phrase mapping table. A phrase alignment tool like `mgiza`[1] can be used for this. Next we augment target monolingual sentences if phrases in that sentence is present in the phrase table and replace that phrase with the corresponding source phrase. This essentially creates a code mixed source sentence. We hope that this could help bring the word embedding of similar words in different languages to be aligned. The augmented data generated by PhraseOut is concatenated with the original parallel data for further training or finetuning.

[1] https://github.com/moses-smt/mgiza

initialization;
$(\mathcal{X}, \mathcal{Y})$: Available parallel data ;
$\tilde{\mathcal{Y}}$: Monolingual data in the target domain ;
Learn a phrase table $\mathcal{P}$ using $(\mathcal{X}, \mathcal{Y})$;
foreach *sentence* $\tilde{\mathbf{y}}$ *in* $\tilde{\mathcal{Y}}$ **do**
  Let $\mathcal{N}$ denote all $ngrams$ in $\tilde{\mathbf{y}}$ (n<=4) ;
  foreach $\tilde{n}_y \in \mathcal{N}$ *chosen randomly* **do**
    if $\tilde{n}_y \in \mathcal{P}$ **then**
      Find corresponding source phrase $\tilde{n}_x$ from $\mathcal{P}$;
      $\tilde{\mathbf{x}} \leftarrow$ Replace $\tilde{n}_y$ with $\tilde{n}_x$ in $\tilde{\mathbf{y}}$
      Append $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ to $(\mathcal{X}, \mathcal{Y})$;
      break
    end
  end
end

Algorithm 1: PhraseOut is a data augmentation method suited for a multilingual neural machine translation

4 Experimental Setup

4.1 Datasets

We experiment on *many to one* multilingual translation from Indian languages to English. We use the WAT 2018 dataset (Nakazawa et al., 2018) for our experiments. It has English translations of several Indian languages. We chose Hindi (hi), Bengali (bn), Malayalam (ml), Tamil (ta) and Telugu (te). The training data size is shown in 1

We use English sentences from the book corpus, a subset of the IIT-B dataset (Kunchukuttan et al., 2018) as our monolingual corpus.

We use moses tokenizer [2] to tokenize English and Indic nlp library [3] for tokenizing Indian languages.

	Train Size	Dev Size	Test Size
bn-en	362,240	1250	1750
hi-en	125,953	1500	2000
ml-en	395,047	1500	2000
ta-en	66,537	1500	2000
te-en	68,573	1500	2000
iitb	-	-	2507

Table 1: Dataset Description

[2] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/tokenizer.perl

[3] https://github.com/anoopkunchukuttan/indic_nlp_library

4.2 Phrase Table Generation

We use Moses (Koehn et al., 2007) to learn phrase tables. The phrase alignment is learned by mgiza and the phrase tables are generated by moses [4]. Bad phrase table entries are removed by filtering by a probability threshold to ensures that good quality phrase translations are extracted out. In order to filter, we multiply the phrase translation probability in both directions and lexical weighting probability (Koehn et al., 2003) in both the directions together and keep the entry only if the multiplied probability is above $1e-12$. A snippet of such a phrase table is shown in 2.

Peter speaking softly	पीटर धीर बोले
Peter taught me	पीटर ने मुझे सिखाया
Peter	पीटर
Petersburg , Russia	पीटर्सबर्ग , रूस
Petersburg ,	पीटर्सबर्ग
Peth .	पेठ ।
Peth	पेठ
Petition ?	याचीका ?
Petition	याचीका

Table 2: A snippet from the phrase table learned using the parallel corpus

4.3 Models and Experimental Procedures

We use the transformer architecture from the fairseq framework [5]. and take an ensemble of last 10 checkpoints for testing.

We use SentencePiece (Kudo, 2018) subword tokenization. A joint subword vocabulary of 16,000 is used for the source side Indian languages and 8,000 is used for the target side English.

5 Results

We provide evaluations that compare several mono lingual augmentation methods, the effect of monolingual data size and the comparison with back-translation.

5.1 Copying vs PhraseOut

We use 200K monolingual sentences and compare mono-lingual augmentation (Copying) against PhraseOut. Note that PhraseOut is applied to augment all 5 language pairs. As shown in the table, PhraseOut gives around 1 BLEU point improvement Copying.

[4] http://www.statmt.org/moses/

[5] https://github.com/pytorch/fairseq

Lang	Baseline	Copying	Switch Out	PhraseOut
wat-hi	30.30	31.14	30.51	**32.06**
wat-bn	20.39	20.86	20.44	**21.07**
wat-ml	17.68	19.04	17.66	**20.62**
wat-ta	20.99	21.60	21.59	**21.64**
wat-te	27.74	28.44	28.36	**29.76**
iitb-hi	8.70	9.52	9.32	**9.90**

Table 3: Results on the WAT 2018 Test Set (Tokenized BLEU score)

5.2 Effect of Monolingual Data Size

We vary monolingual data size from 50K, 100K, 200K to 500K used for PhraseOut. The results on the IITB test set for the Hindi to English translation is shown below.

Size	Baseline	50K	100K	200K	500K
iitb-hi	8.70	9.25	9.68	9.90	10.09

Table 4: Results on IITB Test Set: Monolingual data size vs BLEU score

As shown in the table, the performance of PhraseOut increases with more monolingual data, but the improvement becomes lesser as monolingual data size is further increased.

5.3 Back Translation for Data Augmentation

Back-translation requires a reasonably good model to begin with, since generating too poor synthetic sentences could even deteriorate performance. With the amount of training data we use, BT with NMT doesn't produce good synthetic source sentences. Hence we train SMT (moses) in the reverse direction (i.e. English to Hindi) using the WAT2018 hi-en parallel data and agument 50K back-translated monolingual sentences to the parallel data. We obtain a BLEU score of 8.78 which is only slightly better than the baseline.

5.4 Qualitative Analysis: Translation of Code Mixed Text

Code mixed text is abundant in social media, especially in non-native English speaking countries such as India. A qualitative comparison of translation of a code mixed text is shown in Table 5.

The baseline multi-lingual NMT system is brittle and breaks when it has to translate a code mixed text. On the other hand, a multi-lingual system trained with PhraseOut augmentation is more robust to code mixed input and outputs a reasonable translation.

Source	I am sure minister ने अपने hired writer को बोला होगा "कुछ अच्छा लिखो"
Reference	I am sure the minister would have told his hired writer to ”Write Something Good”
Multilingual NMT	I am sure minister hired writer ” ”
PhraseOut	I am sure minister has told his hired speech to write the good note

Table 5: Translating social media text

6 Related Works

Recently a few works have explored the utility of code mixed (also called code switched) augmentations. (Song et al., 2019) proposes word replacements as in PhraseOut for the purpose of lexicon induction. They perform data augmentation on the parallel data and don't use any monolingual data. Recently code switched pretraining (Yang et al., 2020) (Lin et al.) has shown to work favorably against popular cross lingual pretraining methods.

But all these methods use unsupervised dictionary induction (Artetxe et al., 2018) to obtain parallel word translations. But unsupervised dictionary induction performs quite poorly for distant language pairs such as English to Indian languages (Khatri et al., 2020).

7 Conclusion and Future Work

We propose a simple yet useful data augmentation technique suitable for a multilingual NMT setting, called PhraseOut. Our experiments confirm that PhraseOut is effective in improving the performance of multilingual NMT systems. The improvement in performance could be attributed to better regularization from code mixing.

Training using code mixed data could be useful for improving the robustness of NMT systems. Social media text usually contains code mixed data. One future direction of research is to see how PhraseOut can improve robustness in this kind of a setting.

References

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. In *NAACL*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *ACL*.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. 2019. Mixmatch: A holistic approach to semi-supervised learning. *arXiv preprint arXiv:1905.02249*.

Franck Burlot and Francois Yvon. 2018. Using monolingual data in neural machine translation: a systematic study. In *WMT*.

Anna Currey, Antonio Valerio Miceli Barone, and Kenneth Heafield. 2017. Copied monolingual data improves low-resource neural machine translation. In *WMT*.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *EMNLP*.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *NIPS*.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2020. Revisiting self-training for neural sequence generation. In *ICLR*.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. In *ACL*.

Jyotsana Khatri, Rudra Murthy, and Pushpak Bhattacharyya. 2020. A study of efficacy of cross-lingual word embeddings for indian languages. In *CoDS COMAD*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *NAACL*.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidatesn. In *ACL*.

Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. 2018. The IIT Bombay English-Hindi Parallel Corpus. In *LREC*.

Zehui Lin, Xiao Pan, Mingxuan Wang, Xipeng Qiu, Jiangtao Feng, Hao Zhou, and Lei Li. Pre-training multilingual neural machine translation by leveraging alignment information.

Toshiaki Nakazawa, Katsuhito Sudoh, Shohei Higashiyama, Chenchen Ding, Raj Dabre, Hideya Mino, Isao Goto, Win Pa Pa, Anoop Kunchukuttan, and Sadao Kurohashi. 2018. Overview of the 5th workshop on asian translation. In *WAT*.

Graham Neubig and Junjie Hu. 2018. Rapid adaptation of neural machine translation to new languages. In *EMNLP*.

Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. 2019. Specaugment: A simple data augmentation method for automatic speech recognition. In *arXiv:1904.08779v1*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh neural machine translation systems for wmt 16. In *WMT*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Improving neural machine translation models with monolingual data. In *ACL*.

Kai Song, Yue Zhang, Heng Yu, Weihua Luo, Kun Wang, and Min Zhang. 2019. Code-switching for enhancing nmt with pre-specified translation. In *NAACL*.

Vaibhav, Sumeet Singh, Craig Stewart, and Graham Neubig. 2019. Improving robustness of machine translation with synthetic noise. In *NAACL*.

P Vincent, Hugo Larochelle, Yoshua Bengio, , and P.A. Manzago. 2008. Extracting and composing robust features with denoising autoencoders. In *ICML*.

Xinyi Wang, Hieu Pham, Zihang Dai1, and Graham Neubig. 2018. Switchout: an efficient data augmentation algorithm for neural machine translation. In *EMNLP*.

Zhen Yang, Bojie Hu, Ambyera Han, Shen Huang, and Qi Ju. 2020. Csp: Code-switching pre-training for neural machine translation. In *EMNLP*.

CLPLM: Character Level Pretrained Language Model for Extracting Support Phrases for Sentiment Labels

Raj Ratn Pranesh
Birla Institute of Technology
Mesra, India
raj.ratn18@gmail.com

Sumit Kumar
Birla Institute of Technology
Mesra, India
sumit.atlancey@gmail.com

Ambesh Shekhar
Birla Institute of Technology
Mesra, India
ambesh.sinha@gmail.com

Abstract

In this paper, we have designed a character-level pre-trained language model for extracting support phrases from tweets based on the sentiment label. We also propose a character-level ensemble model designed by properly blending Pre-trained Contextual Embeddings (PCE) models- RoBERTa, BERT, and ALBERT along with Neural network models- RNN, CNN and WaveNet at different stages of the model. For a given tweet and associated sentiment label, our model predicts the span of phrases in a tweet that prompts the particular sentiment in the tweet. In our experiments, we have explored various model architectures and configuration for both single as well as ensemble models. We performed a systematic comparative analysis of all the model's performance based on the Jaccard score obtained. The best performing ensemble model obtained the highest Jaccard scores of 73.5, giving it a relative improvement of 2.4% over the best performing single RoBERTa based character-level model, at 71.5(Jaccard score).

1 Introduction

Sentiment analysis has been a trendy topic for the last some decades. Whether its a graphical image or textual data, all types of an entity consists of something that conveys the sentiment.

With the recent development in machine-learning methods, new innovative and powerful models have developed in the field on natural language processing such heavy pretrained language models.

We designed a novel character-level pretrained language model framework which utilizes the transformers and character-level language models to extract sentiment phrases. The proposed model works in four steps- (i) token-level span prediction using transformer model, (ii) converting token-level representation to character-level representation, (iii) precise character-level span prediction using character-level neural network model, and (iv) retrieving selected sentiment phrases. We also proposed an ensemble character-level model that surpassed the single transformer models. Despite of being a simple idea, ensemble learning has always been successful in several tasks(Zhang and Ma, 2012). We conducted extensive performance analysis of various models and systematically presented our results in the paper.

The primary challenge was in the dataset. As shown in the table 2, for the word such as *likeeee*, the selected sentiment phrase can be *like* or *likee* or *likeee* or *likeeee*. So, by predicting the start and end tokens would not get us to the exact sentiment text span. So, we needed to consider the start and end characters to predict the correct sentiment phrase span. It motivated us to utilize the strong contextual understating of transformers and precise character-level text processing of character-level neural network models for span detection based on different sentiments. Through this study we aim at generating some insights about what exactly a person was thinking while generating any textual content. For example, if a user complains or trolls about a product then there is a need to understand the core reason that dissatisfies that customer about the product. Therefore extracting the phrases that trigger the sentiment can play a significant role in better understanding of the user-generated content.

2 Proposed Transformer Language Models

In this section, we have sequentially discussed and elaborated on the design of the proposed character level pre-trained language models. We first talked about the architecture and working of single Pre-trained Contextual Embeddings (PCE) based character model. Then we talked about the construction

Proceedings of the 17th International Conference on Natural Language Processing, pages 475–480
Patna, India, December 18 - 21, 2020.

and functioning of the character level ensemble model.

2.1 Character Level Transformer Model

The architecture of a character level pre-trained language model divided into two levels. In the experiment, we used 5-fold cross-validation(4-fold data for training and 1-fold for testing) for each PCE model for comparative analysis, but the overall design and workflow were usual in every single PCE model. So we have provided a generalized pipeline that is compatible with all PCE models.

2.1.1 Level 1

At phase 1, we will discuss the transformer's architecture and training procedure along with the conversion of token level predictions to character level predictions. Following are the steps involved in phase 1:

Language Model Processing: We separately used following Pre-trained Contextual Embedding (PCE) models to build Level 1 model: BERT-base-uncased(Devlin et al., 2018a), BERT-large-uncased-WWM(Devlin et al., 2018a), ALBERT-large-v2, ALBERT-base-v2(Lan et al., 2019), RoBERTa-base(Liu et al., 2019) and RoBERTa-large. All the pre-trained models were fine-tuned[1] on SQuAD2.0(Rajpurkar et al., 2018) dataset. The training data was consist of two components-tweet text(t) and sentiment label(s). For an input in the PCE model, the data was structured by concatenating t and s with separator token and classification token added at the beginning. So for BERT and ALBERT the input sequence was `[CLS]`s`[SEP]`t`[SEP]` whereas for RoBERTa it was `<s>`s`</s></s>`t`</s>`. We extracted $AvgPool$ by performing average pooling and $MaxPool$ by performing max-pooling over the output values from hidden states at the same index of each (n-1) layer(except the embedding layer). The $AvgPool$ and $MaxPool$ were concatenated together to form a combined linear vector which was then passed to two fully connected layers, one of size 1024 with tanh activation and next one of size 128. Followed by a multi-sample dropout(Inoue, 2019) layer and finally a softmax activation layer. The model outputs the probabilities of tokens for being start and end of the sentiment phrases span in the tweets. Custom loss as described here 3.3 was used by modifying the cross-entropy loss. The Jaccard score was calculated on test data to measure the performance of the trained model using the test data as described here 3.1.

Character Level Prediction: Once the transformer finished training we had five train models weight because of 5-fold cross-validation. We took the mean start/end token level predictions of all five models on the whole dataset(train+test)b. To make the token level predictions compatible with character-level neural network model we converted the token level start/end predictions to character level start/end predictions. This is done by firstly removing the padding and sentiment label tokens and then assigning each of the characters in the tweet their respective token probabilities received from the transformer(PCE) model.

2.1.2 Level 2

Character Level Neural Network: We designed three character-level neural network models for processing the character level probabilities, and during the experiment3 each model was tested separately. As shown in figure 1, all three models take three inputs: start/end char probabilities, character stream(character level tokens of input tweet), and sentiment label. The models working are described sequentially. (i) In recurrent neural network (RNN) model, we parallelly passed the start/end char probabilities(2 features) through a BiLSTM layer of size 32 along with the characters and sentiment label were through separate embedding layers of each size 32. The three outputs were then concatenated and passed through two BiLSTM layers with a size of 64 each. The output from two BiLSTM were then concatenated with skip connection and passed through a fully connected layer(size=64) and multi-sample dropout(Inoue, 2019)(p=0.5). Finally, a softmax layer that outputs the character level start and end probabilities. (ii) In convolutional neural networks(CNN) model the start/end char probabilities(2 features) were passed into a 1D convolution layer with batch normalization and the characters and sentiment label were through separate embedding layers of each size 32. All three outputs were concatenated and passed through four 1D convolution layer of size 64 with batch normalization, followed by a fully connected layer(size=64) and multi-sample dropout(Inoue, 2019)(p=0.5) and finally a softmax layer. (iii) In the WaveNet

[1]Pre-trained models are available at https://huggingface.co/models

model, similar to CNN model the start/end probability vector, character vector and sentiment vector after concatenation were passed through three WaveBlocks(size=64) with batch normalization, followed by a fully connected layer(size=64) and multi-sample dropout(Inoue, 2019)(p=0.5) and finally a softmax layer.

Retrieving Prediction: The final step in extracting sentiment phrases from the tweet was to convert the character-level start/end probabilities received from the character-level neural network model into the start/end indexes which represent the span of the selected phrases. Once we have the `start index` and `end index`, our model outputs the selected phrases between the `start index` and `end index`. During training, we found that sometimes the `start index > end index`, means that our model was unable to extract any phrases so in output we return the entire tweet as the selected phrase. During training custom loss 3.3was used, and Jaccard scores3.1 was calculated over test data.

2.2 Character Level Ensemble Model

For building the ensemble model, we stacked together with a set of pre-trained language models and character-level neural network models together blended in such a manner that it outperformed all the baseline single character-level model. As shown in figure2 ensemble design is similar to character level transformer model and divided into following two levels:

2.2.1 Level 1

At level 1, we trained each character level transformer model separately following the prior method and stacked the predictions to form a single start/end character level prediction. Following were the steps:

Ensemble Language Model Processing: After training each character level transformer model using 5-fold cross-validation, we had five trained model weights from each transformer model. We predicted a mean token level start/end probabilities using all five model weights and stored them.

Character Level Prediction: As described in the above character level transformer model, we converted the stored token level start/end probabilities from each transformer model into character level start/end probabilities which were then stacked and concatenated together to form character-level ensemble probabilities($Char_{E1}$). While concatenating, we made sure that the probabilities corresponding to the same tweet were combined together.

2.2.2 Level 2

At level 2, the character-level neural network ensemble comes in play. The character-level ensemble model was trained using a 5-fold cross-validation method. Following are the steps involved in level 2:

Character Level Ensemble Neural Network: At this step, we created an ensemble of character-level neural networks using three models: RNN, CNN, and WaveNet. Each of the three character level neural network parallelly receives three inputs: (i) character-level ensemble start/end probability($Char_{E1}$), (ii) character stream, and (iii) sentiment label. Each outputs the character-level start/end probabilities which were then again concatenated and averaged to form a final character-level ensemble start/end probability($Char_{E2}$)

Retrieving Prediction: As discussed above 2.1.2, the start/end probability($Char_{E2}$) was used to extract the sentiment phrases from the tweet. Custom loss and Jaccard scores were used here.

3 Experiment

In this section, we have discussed the dataset, and it's preprocessing step along with the evaluation matrix used in our experiment. We have sequentially explained the experiment setup and model configuration in detail.

3.1 Evaluation Methods

For the evaluation of our proposed single and ensemble character-level pre-trained language models, we used the **Jaccard score** as evaluation metrics. For a given model in training, in each validation fold, $Jaccardscore$ was calculated using the predicted string and ground-truth string and then at the end of the training mean Jaccard score $Jaccard\ score_{\text{mean}}$ of k validation fold was calculated which was used as the final model score.

$$Jaccard\ score = \frac{1}{n}\sum_{i=1}^{n} jaccard(gt_i, dt_i)$$

where, n is the number of tweets in a set, gt_i is the i^{th} ground truth and dt_i is the i^{th} predicted value.

3.2 Dataset

We used Tweet Sentiment Extraction competition dataset[2] publicly available on the Kaggle[3] website. The dataset is comprised of three parts2- (i) tweets (ii) one of the three sentiment classes(Positive, Negative, and Neutral) associated with each tweet (iii) the phrases/words extracted from each tweet that support the sentiment label in the tweet. The total number of tweets in the dataset were 27,481, consisting of 10,992 neutral tweets/8,244 positive tweets/8,245 negative tweets. For the experiment, we used 5 fold stratified cross-validation in which 4 folds i.e. 80%(21,985 tweets) data was for training and 1 fold i.e. 20%(5,496 tweets) data used for validation.

3.3 Experiment Setting

Model training: At level 1: (i) each model was trained separately using 5-fold cross-validation method for five epochs with the batch size of 64(BERT-base, RoBERTa-base), 32(ALBERT-large, Distil-RoBERTa-base) and 16(BERT-large-WWM, RoBERTa-large), (ii) the tokenized input sequence was truncated or padded up to the max length of 100 and Adam(Kingma and Ba, 2014) optimizer was used with learning rate = 3e-5 and weight decay = 0.001. At level 2: (i) character-level model were trained using 5-fold cross-validation for five epochs, (ii) each model had following configurations: max sequence length = 150, training batch size = 128, validation batch size of 512. Trained for five epochs with learning rate of 5e-3, (iii) character-level ensemble model was trained using 5 fold cross-validation for five epochs with the batch size = 8, learning rate = 5e-4 and character-level model configuration was same.

Custom loss (Jaccard-based Soft Labels): Since Cross-Entropy does not optimize Jaccard directly, we tried different loss functions to penalize far predictions more than close ones. We developed a custom loss function that modifies cross-entropy label smoothing by computing Jaccard on the token level. We then use this new target labels and optimize Kullback–Leibler (KL) divergence(Kullback and Leibler, 1951). Alpha here is a parameter to balance between usual cross-entropy and Jaccard-based labelling. On top of this, we used Stochastic Weight Averaging (SWA)(Izmailov et al., 2018) for better generalization to improve the training stability.

Regularization setting: At level 1: Based on the experiments, the best regularization parameters for every architecture were selected. For improving generalization and accelerating training, we used Multi-Sample Dropout(Inoue, 2019)(MSD). Each Transformer had an MSD layer with a probability of 0.5 except for RoBERTa-large, which was 0.6. We add the Gaussian Noise, with = 0.02, to the output layer of the transformers. Based on BERT-paper(Devlin et al., 2018b) we assigned attention probability dropout(0.1) to every transformer. At level 2: As discussed here 2.1.2, for each character-level NN model in the ensemble we used MSD(Inoue, 2019) with the dropout probability value = 0.5.

4 Result and Discussion

In this section, we have discussed the experiment results and presented a performance analysis of character-level transformer models and ensemble models. During our experiment, we combined the level 1 model, with every level 2 model pairwise. As a result, we found out that the RNN based model surpassed other models by achieving a higher Jaccard score during testing. As a result, table1 contains the $Jaccard\ score_{\text{mean}}$ of each transformer combined with the RNN model as Level 2 character-NN model.

According to the table 1, among various single transformer models, the RoBERTa-large model shows good results by having a $Jaccard\ score_{\text{mean}}$ of 71.5%. RoBERTa-base achieved a score of 71.4%, which was very close to RoBERTa-large. Other models also shows promising results, like BERT-large-uncased-WWM and ALBERT-large-V2 had equal $Jaccard\ score_{\text{mean}}$ of 71.1%, whereas the ALBERT-base-V2 and BERT-base-uncased achieved the $Jaccard\ score_{\text{mean}}$ of 70.5% and 71.0% respectively.

Our proposed *ensemble model* outperformed all the single transformer by an average of 2.4%. With a perfect blending of transformer models, the *ensemble model* was able to achieve a $Jaccard\ score_{\text{mean}}$ of 73.5%.

5 Conclusion

In this paper, we proposed a character-level language model consisting of a pre-trained language

[2]Dataset is available at https://www.kaggle.com/c/tweet-sentiment-extraction/data

[3]https://www.kaggle.com/

model and character-level neural network for extracting sentiment support phrases from human-generated tweets based on the associated sentiment labels. We also designed a stacking ensemble model by carefully blending multiple PCE transformer models like BERT, RoBERTa, ALBERT, and character-level neural network models like CNN, RNN, and WaveNet. We conducted a comparative performance analysis of all character-level transformer models and ensemble models. We found that with the optimal combination of transformer models and character-level neural network models, the ensemble architecture generalizes better and outperforms the single transformer models at every level. The ensemble model showed promising results, having a Jaccard Score of 73.5%, which is better than any single transformer model.

Models	*Jaccard Score*$_{mean}$
BERT-base-uncased	71.0
BERT-large-uncased-WWM	71.1
ALBERT-base-V2	70.5
ALBERT-large-V2	71.1
RoBERTa-base	71.4
RoBERTa-large	71.5
Ensemble-Model	**73.5**

Table 1: Models' Scores(in %) on Test Data

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018a. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018b. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Hiroshi Inoue. 2019. Multi-sample dropout for accelerated training and better generalization. *arXiv preprint arXiv:1905.09788*.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Cha Zhang and Yunqian Ma. 2012. *Ensemble machine learning: methods and applications*. Springer.

6 Appendix

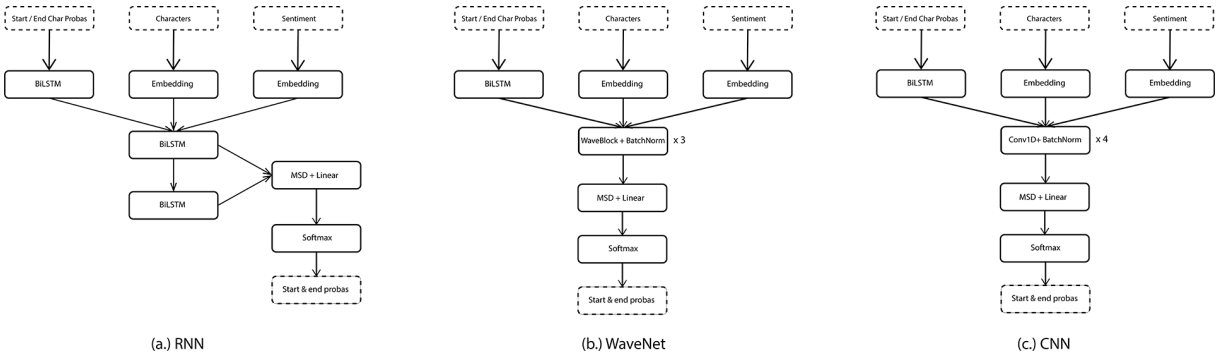


Figure 1: Level 2 Character Level Neural Network (a)Recurrent Neural Network (b)WaveNet (c)Convolution Neural Network

text(given)	**sentiment(given)**	**selected_text(target)**
I really really likeeee the song Love Story by Taylor Swift	positive	likee
I need to get my computer fixed	neutral	I need to get my computer fixed
Sooo SAD I will miss you here in San Diego	negative	Sooo SAD

Table 2: Dataset: texts(given) represents tweets, sentiment(given) represents associated sentiment, selected_text(target) represents extracted phrases.

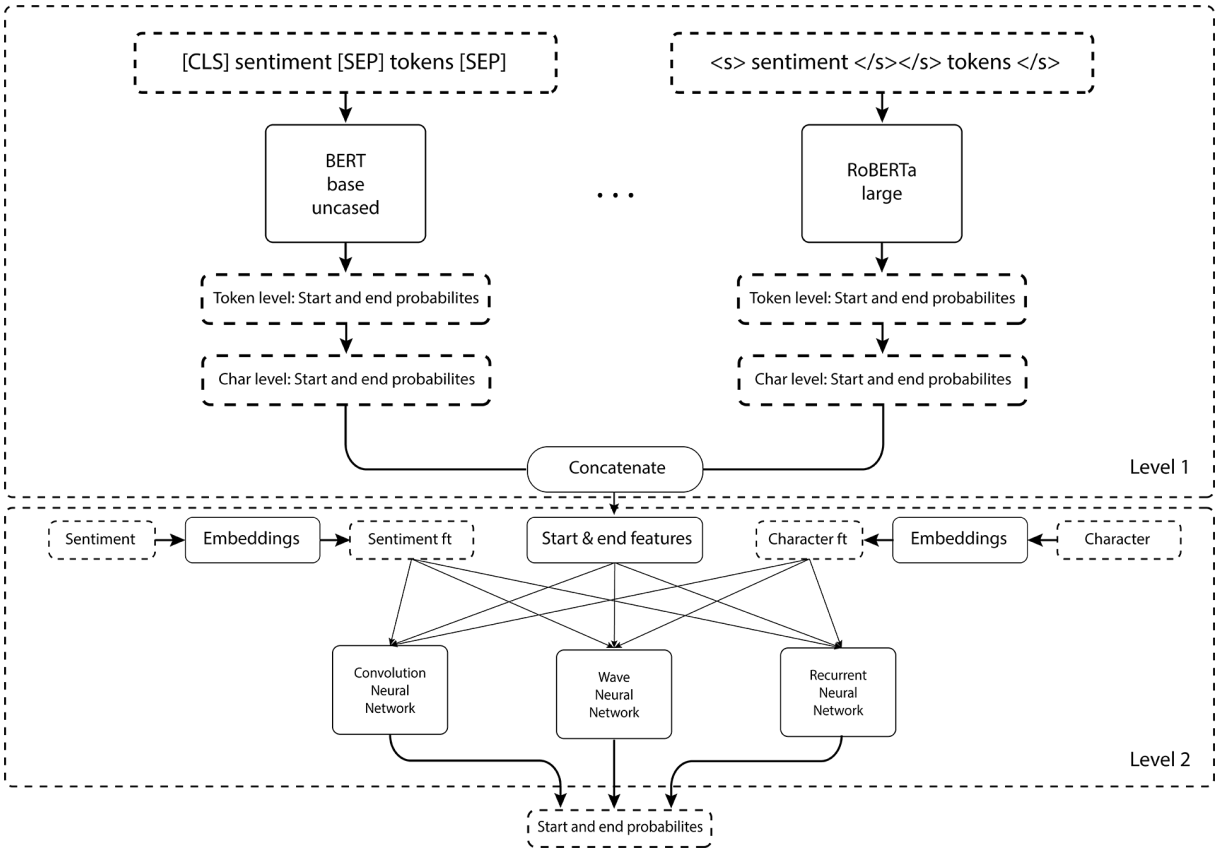


Figure 2: The proposed architecture: Ensemble Model. In level 1, all the transformer models are placed parallelly. In level 2, char-NN models are placed parallelly.

Developing a Faroese PoS-tagging solution using Icelandic methods

Hinrik Hafsteinsson and Anton Karl Ingason
University of Iceland
Reykjavík, Iceland
{hih43,antoni}@hi.is

Abstract

We describe the development of a dedicated, high-accuracy part-of-speech (PoS) tagging solution for Faroese, a North Germanic language with about 50,000 speakers. To achieve this, a state-of-the-art neural PoS tagger for Icelandic, *ABLTagger*, was trained on a 100,000 word PoS-tagged corpus for Faroese, standardised with methods previously applied to Icelandic corpora. This tagger was supplemented with a novel Experimental Database of Faroese Inflection (EDFM), which contains morphological information on 67,488 Faroese words with about one million inflectional forms. This approach produced a PoS-tagging model for Faroese which achieves a 91.40% overall accuracy when evaluated with 10-fold cross validation, which is currently the highest reported accuracy for a dedicated Faroese PoS-tagger. The tagging model, morphological database, proposed revised PoS tagset for Faroese as well as a revised and standardised PoS tagged corpus are all presented as products of this project and are made available for use in further research in Faroese language technology.

1 Introduction

This paper describes the developement of a high-accuracy Part-of-Speech (PoS) tagging solution for Faroese, a North Germanic spoken by about 50,000 people in the Faroe Islands, an autonomous territory in the Kingdom of Denmark. Limited research has been performed on such an implementation for the language and as PoS taggers are fundamental in various further implementations in natural language processing (NLP) and linguistic research, the need for new research is apparent. In the current project, as a basis for a Faroese PoS tagger, a state-of-the-art bi-LSTM neural PoS-tagging system for Icelandic, *ABLTagger* (Steingrímsson et al., 2019), is used as a foundation to build on, in addition to various methods used in Icelandic NLP research. The reason why these Icelandic resources may also be applied to Faroese is the extensive grammatical similarities between the two languages. These similarities are especially apparent in morphology, as both languages retain grammatical categories and nuances not apparent in related languages.

In contrast with Faroese, the last two decades have seen broad gains for Icelandic in the field of language technology (LT), producing tools and databases which have enabled both various new technical implementations for the language and new opportunities in linguistic research. With the grammatical similarities of the two languages in mind, similar gains should be possible for Faroese.

Using Icelandic NLP tools and methods, a Faroese PoS-tagging model was produced which achieves an overall tagging accuracy of 91.40%. This is considerably higher than the previous dedicated PoS tagger for Faroese which achieved 87.00% using a similar tagset and trained on the same corpus, the Faroese Sosialurin corpus, which contains news articles, totalling about 100,000 words (Hansen et al., 2004). Furthermore, the current project produced various data set innovations which could prove useful in further NLP research for Faroese. These include a proposed revised tagging scheme for Faroese which is optimised for high-accuracy PoS-tagging and a standardised version of hand-corrected PoS tagged corpus for use in NLP projects. In addition to this, the project produced a novel inflection database for Faroese, the Experimental Database of Faroese Morphology (EDFM), which contains detailed morphological information on 67,488 Faroese words, modelled on the Dictionary of Icelandic Morphology (Bjarnadóttir, 2012; Bjarnadóttir et al., 2019). These data sets have been made available online[1] for fur-

[1] See: https://github.com/hinrikur/far-ABLTagger.

Proceedings of the 17th International Conference on Natural Language Processing, pages 481–490
Patna, India, December 18 - 21, 2020.

ther development and research in Faroese language technology.

Section 2 outlines the ABLTagger PoS-tagging system and previous work on Faroese LT relevant to the current project, along with discussing the applicability and incentives to use Icelandic materials and methods to develop LT solutions for Faroese. Section 3 describes the collection and preparation of the materials used to implement ABLTagger for Faroese. Section 4 describes the training and evaluation of the Faroese ABLTagger model and Section 5 discusses the current applicability of the Faroese PoS tagging model and the next steps in improving it as a whole. Section 6 concludes.

2 Background

2.1 The ABLTagger system

The current project draws inspiration from the ABLTagger experiment for Icelandic (Steingrímsson et al., 2019), which produced a Bi-directional LSTM PoS tagger which uses a morphological database to achieve high accuracy tagging of the language, using a fine-grained tagset. When trained on a hand-corrected corpus, the *IFD* corpus (Pind et al., 1991), a total of about 500,000 words, the ABLTagger system achieved an overall tagging accuracy of 94.17%, making it the state-of-the-art PoS-tagging implementation for Icelandic.

The ABLTagger system for Icelandic uses a fine-grained tagset of about 600 PoS tags, originally introduced in the aforementioned IFD corpus and revised in further research, notably in the MIM-GOLD corpus (Barkarson et al., 2020). An excerpt of this tagging scheme is shown in Table 1. In the tagging scheme, each token receives one tag string. Each tag string contains a series of symbols, each containing important grammatical information on the token, e.g., case, number, tense and grammatical gender.

The morphological component of the ABLTagger system consists of a morphological database for the language being tagged. For Icelandic, Steingrímsson et al. used the Databse of Icelandic Morphology, *DIM* (Bjarnadóttir, 2012; Bjarnadóttir et al., 2019), in the so-called DIM basic format.[2] *DIM* contains around 290,000 inflectional paradigms with over 5.8 million inflectional forms, aiming to be a descriptive resource for Icelandic. This database is freely available under a CC BY-SA 4.0 license in standardised formats and is for the most part manually corrected.

Providing these two components of the ABLTagger system, the PoS-tagged corpus and morphological database, is essential for implementing the system for a new language and is the main focus of the current project.

2.2 Previous work on Faroese

Although extensive research in Faroese LT is scant, some research has taken place in the past decades. Experiments have been performed in machine-parsing Faroese text using transfer learning with Icelandic data (Ingason et al., 2014) and work on a finite-state based grammatical analyser for Faroese is ongoing (Trosterud, 2009).

Notably, the Sosialurin corpus project (Hansen et al., 2004) consisted of the formulation of a fine-grained PoS tagging scheme for Faroese and the compilation of a hand-corrected, PoS-tagged text corpus using the tagset. This corpus was used to train the TnT tagger (Brants, 2000), which achieved an overall tagging accuracy of 87.0% at the time. Furthermore, the Sosialurin corpus and further machine-tagged text have been made accessible for linguistic research on the CorpusEye (Bick et al., 2020) website.[3] These resources, the Sosialurin corpus and its corresponding tagging scheme, are instrumental for the current project, as they provide the training material for the ABLTagger system.

2.3 Applicability of Icelandic methods for Faroese

The foremost incentive for using Icelandic NLP tools for Faroese is the scarcity of NLP implementations for the latter. A handful of previous research exists, but very few extensive databases and ready-to-use software are available for the language at large. As a result, the amount of available digital language resources for Faroese is limited. In contrast, the last two decades have seen great gains for Icelandic in the field of language technology (Nikulásdóttir et al., 2017). This has produced tools and databases which enable both new technical implementations for the language and new opportunities in linguistic research, both of which would also be beneficial for Faroese.

The fundamental reason that makes Icelandic NLP implementations applicable for Faroese are the grammatical similarities between the two lan-

[2] See: https://bin.arnastofnun.is/DMII/LTdata/s-format/

[3] See: https://corp.hum.sdu.dk/.

Token	Lemma	PoS tag	Explanation
Ég	ég	fp1en	**f**: pronoun; **p**: personal; **1**: 1st person; **e**: singular; **n**: nominative;
stökk	stökkva	sfg1eþ	**s**: verb; **f**: indicative; **g**: active; **1**: 1st person; **e**: singular; **þ**: past tense
á	á	aa	**a**: adverb; **a**: doesn't govern case;
eftir	eftir	aþ	**a**: adverb; **þ**: governs dative;
strætó	strætó	nkeþ	**n**: noun; **k**: masculine; **e**: singular; **þ**: dative;
og	og	c	**c**: conjunction;
veifaði	veifa	sfg1eþ	**s**: verb; **f**: indicative; **g**: active; **1**: 1st person; **e**: singular; **þ**: past tense

Table 1: Excerpt from the *IFD* corpus, with explanations. The IFD corpus was used to train the ABLTagger system for Icelandic by Steingrímsson et al. (2019) and as a basis for the original Faroese tagging scheme by Hansen et al. (2004).

guages. These similarities are especially apparent in morphology, as both languages retain grammatical categories not apparent other similar languages, e.g., four grammatical cases for nominals and an extensive conjugation system for verbs, to name a few. Furthermore, the similarities also extend to the syntax of the languages and orthographies, although with various systematic differences in both. With this in mind it can be supposed that some NLP tools that perform well for Icelandic may also perform well for Faroese, especially data-driven applications.

3 Data collection and preparation

A cornerstone of implementing the ABLTagger system for Faroese is collecting and preparing the resources needed for the task. In the current project, this consisted of standardising the training corpus, revising an already existing tagging scheme for Faroese and compiling an experimental morphological database. This is described in the following section.

3.1 Sosialurin Corpus

The aforementioned Sosialurin project, conducted by Hansen et al. (2004) aimed at gathering text into a sizeable corpus, which would then be machine-tagged and finally manually corrected to create a standardised PoS-tagged corpus for Faroese, to be used in NLP projects and linguistic research. This corpus, hereafter referred to as the Sosialurin corpus, consists of 221 excerpts from the newspaper Sosialurin.[4] In total, the corpus contains 119,833 tokens in 4,073 sentences, about 104,000 words excluding punctuation.

As the corpus was meant to be used in NLP projects and linguistic research in general, the corpus comes prepared in a *token-tag* format, where each line of the corpus file contains a single token and its corresponding PoS tag, separated by a `tab` character. Sentences are demarcated with an empty line and tokens are PoS-tagged using a PoS tagging scheme devised specifically for the corpus.

Before use in the current project, the Sosialurin corpus was slightly modified from the original. This included removing duplicated sentences and metadata from the corpus text and standardising organisation and sentence demarcation. This process produces a revised version of the Sosialurin corpus, bringing the total number of sentences in the revised corpus to 6,156, from the original corpus' 4,073 and the total number of tokens to 117,690 from the original 119,819. It is worth noting this corpus is less than 10% as large as the combined training corpus used to train the original ABLTagger implementation for Icelandic, as discussed in Section 2.1. Nevertheless, as the largest hand-corrected, Faroese PoS tagged corpus, this revised version of the Sosialurin corpus was used to train the ABLTagger implementation for Faroese, with the supposed effect of its relatively small size being discussed further in Section 4.

3.2 Revised Faroese tagging scheme

The tagging scheme devised for the Sosialurin corpus by Hansen et al. (2004) is, to a large extent, based on the tagging scheme used in the IFD corpus for Icelandic (Pind et al., 1991). In the last decades, a number of revisions have been made to the IFD

[4] Accessible at: https://www.sosialurin.fo/.

tagging scheme, mostly to improve tagging efficiency, with the latest version appearing in the most recent release of the MIM-GOLD hand-corrected, PoS-tagged corpus (Barkarson et al., 2020). The same cannot be said about the Sosialurin tagging scheme, as no substantial revisions have been made to it since its inception. As such, before being used to train the ABLTagger system, a number of revisions were applied to the tagging scheme and subsequently to the PoS tags in the Sosialurin corpus itself.

Most of the revisions applied to the Faroese tagging scheme were based on revisions previously applied to the IFD tagging scheme for Icelandic. These include reworked numeral and punctuation tag strings, simplified case governance tagging for adverbs and the removal of a dedicated tag for past participles. Furthermore, various new tag strings were introduced, also based on the IFD tagging scheme, e.g., distinction between different categories of pronouns.

One example of a language-specific revision made on the tagset was the removal of distinction between person (1st, 2nd or 3rd) from verb tags in the original tagset. This was likely a carry-over from the IFD tagging scheme, as Icelandic verbs are morphologically distinct between person in both singular and plural. In Faroese, verbal person is not morphologically apparent in plural forms, and thus should not be relevant to the tagging scheme, in theory. The effect of this revision to the tagging scheme on tagging accuracy is discussed in Section 4.

When applied to the Sosialurin corpus, the total number of unique PoS tags in the corpus was reduced from 390 to 371. This total does not reflect the tagset changes at large, as not all possible tag strings in the tagging scheme are represented in such a small corpus and while many possible tag strings were removed from the tagset, a number of possible tag strings were added as well.

3.3 Experimental Database of Faroese Morphology (EDFM)

The *ABLTagger* system uses a morphological database, a detailed description of the inflection of a language, in tandem with a bi-LSTM-based neural tagger to enhance the accuracy of the tagger as a whole. In the original experiment for Icelandic, the Database of Icelandic Morphology, *DIM* (Bjarnadóttir, 2012; Bjarnadóttir et al., 2019) was used for this purpose, as Icelandic was the language being tagged. As discussed in Section 2.1, *DIM* is an extensive database with hundreds of thousands of unique lemmas and millions of inflected word forms on file, and was applied in the so-called DIM basic format[5] for the project.

No such database, comparable in size, scope or accessibility to *DIM*, exists for Faroese. This situation of course poses a problem when trying to replicate the *ABLTagger* experiment for Faroese, as the morphological component is essential the high accuracy of the tagging mechanism. For the current project, we overcame this by gathering all freely accessible information on Faroese inflection into an Experimental Database of Faroese Morphology (EDFM), formatted in the DIM basic format. This database can then used for Faroese, the same way *DIM* is used in the *ABLTagger* system. As discussed in Section 2.3, Faroese and Icelandic are to large extent morphologically similar, so this approach is at least theoretically applicable.

3.4 Sourcing morphological data

The inflectional data used to build the EDFM was collected from several main sources. In Table 2, the number of paradigms extracted from each data set is shown, also categorised by lexical category.

Faroese Dictionary Database: The largest single morphological description of Faroese is provided by the Dictionary of Faroese (*Føroysk Orðabók*, Poulsen et al. 1997), hereafter referred to as the *OBG* database. This database is accessible and searchable on the website of Sprotin,[6] a Faroese publishing house which provides digital access to various Faroese dictionaries. This database contains 67,488 word entries, of which 65,062 contain inflectional information and were used in the EDFM. Out of these, 5,374 entries (mainly verbs and numerals) contained partial inflectional paradigms that were extended using Python scripts written for the current project.

Faroese naming committee: Along with the OBG database, the Sprotin website hosts a complete list of approved given names in Faroese, with each name's inflection included. Faroese naming laws dictate that only given names that are approved by a governmental naming committee

[5] See: https://bin.arnastofnun.is/DMII/LTdata/s-format/

[6] Accessible at: www.sprotin.fo/dictionaries.

Category	OBG	Names	Wiktionary	Generated	Manual	Total
Adjectives	11,907	-	16	-	-	11,923
Adverbs	1,289	-	-	-	-	1,289
Conjunctions	-	-	-	-	61	61
Interjections	-	-	-	-	115	115
Nouns	46,492	1,667	113	-	-	48,272
Numerals	-	-	-	47	57	104
Prepositions	-	-	-	-	62	62
Pronouns	-	-	-	-	20	20
Verbs	-	-	7	5,327	-	5,334
Total	59,688	1,667	136	5,374	315	**67,180**

Table 2: Contents of the EDFM by lexical category and source database.

can be used officially (Faroese Naming Committee, 2020). This list provided 1,667 Fareoese given names to the EDFM, containing 880 masculine given names and 787 feminine.

Wiktionary data: The English-language Wiktionary contains entries for various Faroese words, most of which contain morphological information. These entries the were accessed via morphological data from the UniMorph project (Kirov et al., 2018; McCarthy et al., 2020), which was originally extracted and generated from Wiktionary data dumps, specifically from June 20, 2015 (Kirov et al., 2016) and is freely available online.[7] Although 3,077 in total, 2,687 of the UniMorph entries were already represented in the entries extracted from the OBG database. Nevertheless, 390 new entries were extracted, further improving the EDFM.

Manual paradigms: A total of 315 entries in the EDFM were manually defined using morphological descriptions of Faroese as guidelines, e.g. Þráinsson et al. (2004). These were mostly pronouns and functional words, in addition to a number of uninflectable words.

3.5 Formatting the EDFM

As discussed in Section 2.1, the morphological component of the original *ABLTagger* experiment for Icelandic was applied in the DIM basic format. As such, the Faroese inflectional data had to be standardised in a similar manner, in order to be applied in the *ABLTagger* system. This was achieved automatically with purpose built scripts.

To illustrate the output format of the EDFM, the entry for the Faroese word *grunnur* 'foundation' in the EDFM is shown in Figure 1. The format consists of lines of comma-seperated fields, with each line containing an inflectional form of a given word. These lines are grouped into word entries, identifieable by the **lemma**, here `grunnur`, as shown in the first field, and the **database ID number**, here `18433`, as shown in the second field. The remaining fields are **lexical category**, here `kk` for masculine noun, **source data set**, here `obg` for the OBG database, **inflected form** of the word and finally the **database specific grammatical tag**, which encodes morphological information on the word form.

```
grunnur;18433;kk;obg;grunnur;NFET
grunnur;18433;kk;obg;grunn;ÞFET
grunnur;18433;kk;obg;grunni;ÞGFET
grunnur;18433;kk;obg;gruns;EFET
grunnur;18433;kk;obg;grunnar;NFFT
grunnur;18433;kk;obg;grunnar;ÞFFT
grunnur;18433;kk;obg;grunnum;ÞGFFT
grunnur;18433;kk;obg;grunna;EFFT
grunnur;18433;kk;obg;grunnurin;NFETgr
grunnur;18433;kk;obg;grunnin;ÞFETgr
grunnur;18433;kk;obg;grunninum;ÞGFETgr
grunnur;18433;kk;obg;grunsins;EFETgr
grunnur;18433;kk;obg;grunnarnir;NFFTgr
grunnur;18433;kk;obg;grunnarnar;ÞFFTgr
grunnur;18433;kk;obg;grunnunum;ÞGFFTgr
grunnur;18433;kk;obg;grunnanna;EFFTgr
```

Figure 1: Excerpt from EDFM output in the DIM basic format: Inflections of *grunnur*.

Ideally, each word entry contains the full inflectional paradigm for the given word. In its current experimental form, the EDFM contains mostly full paradigms, although it contains partial paradigms for certain lexical categories. Furthermore, as the database has not been formally proofread, there are bound to be some errors in the data, especially in the automatically generate paradigms. As will be

[7] Accessible at: https://github.com/unimorph/fao.

discussed in Section 4, this does not disqualify the EDFM from use in LT implementations, such as ABLTagger, although revisions remain a focus for further work on the database.

4 Evaluation of tagger

The *ABLTagger* system as described by (Steingrímsson et al., 2019) uses DyNet[8] (Neubig et al., 2017). The architecture and model hyperparameters used in the evaluation were all unchanged from the original *ABLTagger* experiment.[9] This included stochastic gradient descent training with initial learning rate of 0.13, which decays 5% per epoch, running for 30 epochs for the full model. The hidden layer of the network has 32 layers. The input text, PoS tags and morphological data are vectorized before use in the system and the resulting embeddings for words, characters and the morphological component have 128, 20 and 61 dimensions respectively.

In accordance with previous experiments (see, e.g., Loftsson 2006; Barkarson 2018; Ingólfsdóttir et al. 2019; Steingrímsson et al. 2019), training and evaluation was done via 10-fold cross validation. In this approach, the whole data set at hand is used for both training and testing. This is especially useful when the data set is not large enough to effectively split into dedicated training and testing sets.

4.1 Evaluation setup

Three variables were taken into account to evaluate the Faroese ABLTagger implementation. These were the effect of the tagset revisions discussed in Section 3.2, the effects of the size and contents of the training corpus, and the effect of adding the morphological component described in Section 3.3 to the ABLTagger system.

Tagset revisions: Three models were trained, each using the Sosialurin corpus with a specific tagset with varying amounts of revisions:

- **S-Baseline:** The original Faroese tagset by Hansen et al. (2004)
- **S-Revised-V:** The revised Faroese tagset, with unchanged verbal plural tags (see discussion in Section 3.2)
- **S-Revised:** The fully revised Faroese tagset

[8]The Dynamic Neural Network Toolkit, see http://dynet.io.

[9]The *ABLTagger* source code is available at https://github.com/steinst/ABLTagger.

Corpus size and contents: To evaluate the performance of ABLTagger on small corpora in general, three comparison models were trained, using subsets of the hand-corrected *Icelandic* MIM-GOLD corpus, with each subset corpus being of a relatively similar size and text genre as the Sosialurin corpus, i.e. news articles:

- **MIM-F:** Texts from the newspaper *Frét-tablaðið*. **Size:** 94,224 tokens
- **MIM-M:** Texts from the newspaper *Morgunblaðið*. **Size:** 243,346 tokens
- **MIM-MR:** Texts from the newspaper *Morgunblaðið*, resized to same size as Sosialurin. **Size:** 117,957 tokens

Addition of morphological data: A Faroese model trained on Sosialurin, with the tagset which provies the best overall accuracy, along with the EDFM as the morphological component. Additionally, three reference models were trained using the same Icelandic corpora as above, with the DIM as the morphological component:

- **S-Morph:** Sosialurin with optimal tagset + *EDFM* as morphological component
- **MIM-F-Morph:** MIM-F model + *DIM* as morphological component
- **MIM-M-Morph:** MIM-M model + *DIM* as morphological component
- **MIM-MR-Morph:** MIM-MR model + *DIM* as morphological component

4.2 Results

The evaluation results of the three "tagset models", S-Baseline, S-Revised-V and S-Revised, are is shown in Table 3.

Model	Token	Sentence	Known	Unknown
S-Baseline	88.92%	23.50%	91.70%	55.85%
S-Revised-V	89.75%	24.09%	92.63%	55.76%
S-Revised	**90.12%**	**25.55%**	**93.01%**	**56.01%**

Table 3: Accuracy of taggers trained on different tagsets.

It is apparent that the baseline model trained on the original, unrevised tagging scheme achieved the lowest overall accuracy of the three. Adding only the revisions based on the Icelandic MIM-GOLD corpus to the tagset (S-Revised-V), as described in Section 3.2, resulted in an accuracy gain of 0.83%,

	S-Baseline		S-Revised-V		S-Revised	
No.	Proposed tag > correct tag	Error rate	Proposed tag > correct tag	Error rate	Proposed tag > correct tag	Error rate
1.	ED > EA	2.24%	DN > DG	3.07%	DN > DG	3.35%
2.	EA > ED	1.91%	DG > DN	3.05%	DG > DN	3.17%
3.	VNPP3 > VI	1.59%	VI > VNPP3	1.68%	VNPP > VI	2.15%
4.	VI > VNPP3	1.56%	C > CI	1.65%	VI > VNPP	1.84%
5.	C > CI	1.51%	VNPP3 > VI	1.52%	C > CI	1.69%
6.	EA > EN	1.33%	C > CR	1.33%	CI > C	1.20%
7.	EN > EA	1.16%	CI > C	1.25%	CR > C	1.19%
8.	CI > C	1.16%	CR > C	1.15%	C > CR	1.17%
9.	CR > C	1.13%	DN > C	0.83%	DN > C	0.92%
10.	C > CR	0.98%	C > DN	0.72%	C > DN	0.77%

Table 4: 10 most common errors in tagset evaluation, divided by tagging scheme

equivalent to a total error reduction of 7.51%. Further omission of grammatical person in plural verb PoS tags raised the overall accuracy of the model by another 0.23%, pushing the total error reduction to 10.05% compared to the baseline. The fully revised tagging scheme also achieved the highest scores in whole-sentence accuracy and for both known and unknown tag accuracy, without much loss of grammatical information in the tagging scheme. As such, the S-Revised model was used for the further evaluation steps.

Although the fully revised tagging scheme produces the most accurate model out of the three, there are some systematic errors in the tagging that the revisions do not affect. In Table 4, the 10 most common errors in the models are shown. The most common errors in all the models concern adverbs and prepositions, specifically concerning case governance. In Table 4, these are lines 1.-2. for the revised models and 1.-2. and 6.-7. for the baseline model. These errors play a bigger role in the S-Baseline model, as the PoS tags **EN**, **EA**, **ED**, **EG** refer to prepositions that govern *nominative*, *accusative*, *dative* and *genitive* case, respectively, with substantial ambiguity between tokens that receive these tags. In the revised tagset, these are replaced by **DN** and **DG**, for prepositions[10] that *do not govern case* and those which *do govern case*, respectively, eliminating some of this ambiguity. Similar to these are the various errors concerning conjunctions (or complementisers), i.e., the tag strings starting with **C**. This is mainly caused by the words *at*, *ið* and *sum* (all meaning 'that') which can be variously tagged as conjunctions, relative conjunctions or (in the case of *at*) as infinitive markers (in which case it means 'to').

After removing grammatical person from tags of plural verbs, these tags continue to cause errors. However, the errors in question, underlined in Table 4, are not caused by ambiguity within the plural verb tags themselves. These errors are caused by the Faroese verbal infinitive form, which should receive the tag **VI**, being lexically identical to the active present plural form, which should receive the tag **VNPP** after the tagset revisions. Although at first glance, the table may suggest that this type of error has a higher rate of occurrence in the fully revised model (from 1.68% and 1.52% to 2.34% and 1.86%) this is not the case. This is simply because the revised tagset merges the plural tags into **VNPP**, thus "collecting" the errors of this type. At any rate, the omission of grammatical person in plural verb PoS tags raises the overall accuracy of the model by a substantial amount without much loss of grammatical information.

The three models trained on *MIM-GOLD* subcorpora described above were evaluated in the same manner as the tagset models, with the results shown in Table 5. Also shown in the table are the full baseline model (without morphological data) from the original *ABLtagger* experiment for Icelandic and the S-Revised model, the model which achieved the highest accuracy above, along with the total token count of all the training corpora used.

There seems to be a correlation between corpus size and tagging accuracy; the larger the training corpus, the higher the achieved tagging accu-

[10] In the original tagging scheme prepositions receive a tag string starting with **E**. These are merged with adverbs (**D**) in the revised tagging schemes.

Model	Overall	Known	Unknown	Corp. sz.
S-Revised	90.12%	93.01%	56.01%	117,690
MIM-F	87.28%	93.46%	56.75%	94,224
MIM-MR	88.31%	93.22%	59.78%	117,957
MIM-M	91.03%	94.41%	64.85%	243,346
ABLTagger	93.25%	95.19%	66.84%	590,279

Table 5: Revised Sosialurin model, Icelandic reference models and the original ABLTagger implementation (Steingrímsson et al., 2019) *without morphological data*.

racy is. With this in mind, at 90.06%, the Faroese S-Revised model achieves a relatively high accuracy, surpassing the overall accuracy of two of the smaller Icelandic reference corpora, although not approaching the 93.25% of the original *ABLTagger* baseline model. These accuracy scores are not directly comparable, as although the technical aspects of the models are identical (and, in theory, the text genre of the training corpora), the tagset used for Faroese is still simpler than the one for Icelandic, which may affect the final tagging accuracy. The results do however show that while the accuracy of the Faroese S-Revised model is not comparable to the original *ABLTagger* baseline, it is in the same ballpark as the Icelandic reference models.

The evaluation of the *ABLTagger* model supplemented with EDFM is shown in Table 6.

Model	Token	Sentence	Known	Unknown
S-Revised	90.12%	25.55%	**93.01%**	**56.01%**
S-Morph	**91.40%**	**29.01%**	92.89%	51.41%

Table 6: Sosialurin morphology evaluation results.

The full model with morphological data achieves a overall accuracy of 91.40%, the highest for all the Faroese models. When compared to the S-Revised model, which achieved a 90.12% accuracy, this shows that applying the EDFM raises the final accuracy by 1.28%, amounting to a total error reduction of 12.96%.

The comparison of the S-Morph model to the Icelandic reference models is shown in Table 7. Each model's accuracy is shown along with the accuracy gain provided by the morphological data. In comparison to the Icelandic reference models, the accuracy gain that the EDFM provides when tagging Faroese is comparatively low. Indeed, the DIM contains a much more thorough description of Icelandic than the EDFM does of Faroese and thus should in theory provide better results when applied with *ABLTagger*. However, these results show that despite its experimental nature, the EDFM does indeed serve its purpose in raising the overall tagging accuracy of the ABLTagger system.

Model	Accuracy	Morph. gain
S-Morph	91.40%	+1.28%
MIM-F-Morph	90.69%	+3.41%
MIM-MR-Morph	91.85%	+3.54%
MIM-M-Morph	92.36%	+1.33%

Table 7: Morphology model results and accuracy gain from morphological data.

5 Application and further work

It remains to be discussed how effectively the tagger produced in the current project can be applied in PoS-tagging Faroese text in general. In Table 8, the current project's S-Morph model, hereafter referred to as the Faroese *ABLTagger* model, is compared to the the last dedicated PoS-tagging implementation for Faroese, by Hansen et al. (2004), as mentioned in Section 2.2.[11] Although this tagger used the unrevised Faroese tagset, as discussed in Sections 3.1 and 3.2, and the exact evaluation procedure used is not known, it can serve as a tentative comparison for the current project, in lieu of a previous state-of-the-art tagging implementation for Faroese.

Implementation	Overall	Known	Unknown
Hansen et al. (2004)	87.00%	91.00%	64.70%
Faroese *ABLTagger* model	91.40%	92.89%	51.41%

Table 8: Tagging accuracy for the current project compared to previous best

As is apparent in Table 8, with an overall tagging accuracy of 91.40%, the model produced in the current project returns a substantial improvement on the previous tagger, which achieved an accuracy of 87.00%. The result achieved by the Faroese *ABLTagger* model is quite promising, especially as it is uses a quite fine-grained tagset. By these metrics, the current project has produced the most accurate dedicated, fine-grained PoS tagger for Faroese to date.

[11] The Faroese Giellatekno implementations (Trosterud, 2009), although not containing a PoS tagger per se, do contain a rule based grammatical analyser, which can function somewhat like a PoS tagger. However, these implementations have not been evaluated in a similar way to the taggers discussed here and are thus left out of the discussion. The possibility of future comparisons remains.

Despite the high reported accuracy of the Faroese model, two issues must be kept in mind. Firstly, the overall accuracy, although high, does not approach the accuracy of the full *ABLTagger* model for Icelandic, which, as discussed in Section 2.3, has a similar tagset and overall morphology to Faroese. In theory, a substantially higher tagging accuracy should be obtainable for Faroese with the *ABLTagger* system, but it is limited by the size and contents of the training data used. Secondly, as the Sosialurin corpus only consists of news articles, its contents are likely not representative of Faroese-language texts in general. The Faroese *ABLTagger* model would thus perform well on news-like texts in its current form, but likely return sub-optimal results when tagging large, unseen texts in different genres, which is the main goal when developing a PoS tagger.

6 Conclusion

In this paper we have described the developement of a dedicated, high-accuracy PoS-tagging solution for Faroese. This was achieved by training *ABLTagger*, a state-of-the-art PoS-tagging system for Icelandic, on Faroese language data which had been revised and formatted with methods and tools based on Icelandic NLP research. This produced a Faroese PoS-tagging model which achieves a 91.40% overall tagging accuracy, when trained on the 100,000 word Sosialurin corpus and evaluated using 10-fold cross validation. The last similar PoS-tagging implementation for Faroese achieved a 87.0% overall accuracy. Thus, in the absence of recent comparable implementations, the Faroese *ABLTagger* model may tentatively be considered the state-of-the-art for PoS-tagging Faroese.

In addition to developing the PoS tagger, this project produced various resources which could prove useful in further research in Faroese language technology. These include a proposed revised PoS-tagging scheme for Faroese, mainly based on the Icelandic MIM-gold tagging scheme, as well as the standardised and revised version of the corresponding Sosialurin PoS-tagged Corpus. The same goes for the EDFM, the experimental morphological database compiled for use with the tagger. Although its format is based on its Icelandic counterpart, DIM, and is mostly compiled from already existing Faroese dictionary data, with 67,488 word forms and about 1,000,000 inflectional forms, it is the first of its kind for Faroese as a single, accessible data set designed for use in language technology implementations.

As Faroese digital language resources are, at the moment, few and far between, Faroese language technology has ground to cover before it can be considered fully equipped to tackle recent innovations in the field. As the data sets and PoS-tagging model produced in this project have been made available online,[12] they may well serve as a basis for further developments for Fareoese, both to implement new NLP applications and provide further opportunities for linguistic research.

References

Starkaður Barkarson. 2018. Þjálfun málfræðimarkarans Stagger með nýjum gullstaðli [Training of the PoS tagger Stagger with a New Gold Standard]. Unpublished MA thesis. URL http://hdl.handle.net/1946/29474.

Starkaður Barkarson, Einar Freyr Sigurðsson, Eiríkur Rögnvaldsson, Hildur Hafsteinsdóttir, Hrafn Loftsson, Steinþór Steingrímsson, and Þórdís Dröfn Andrésdóttir. 2020. MIM-GOLD 20.05. CLARIN-IS, Stofnun Árna Magnússonar.

Eckhart Bick, Heini Justinussen, Zakaris Svabo Hansen, Trond Trosterud, and Tino Didriksen. 2020. Corpuseye Faroese Corpus.

Kristín Bjarnadóttir. 2012. The database of modern Icelandic inflection (Beygingarlýsing íslensks nútímamáls). In *Proceedings of the Workshop on Language Technology for Normalisation of Less-Resourced Languages (SaLTMiL 8 – AfLaT2012)*, pages 13–20, Istanbul, Turkey. European Language Resources Association.

Kristín Bjarnadóttir, Kristín Ingibjörg Hlynsdóttir, and Steinþór Steingrímsson. 2019. Dim: The database of Icelandic Morphology. In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pages 146–154, Turku, Finland. Linköping University Electronic Press.

Thorsten Brants. 2000. TnT: A statistical Part-of-Speech Tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*, ANLC '00, page 224–231, USA. Association for Computational Linguistics.

Faroese Naming Committee. 2020. Góðkend Fólkanøvn [Approved Given Names].

Zakaris Svabo Hansen, Heini Justinussen, and Mortan Ólason. 2004. Marking av teldutøkum tekstsavni [Tagging of a digital text corpus].

[12] See: https://github.com/hinrikur/far-ABLTagger.

Anton Karl Ingason, Hrafn Loftsson, Eiríkur Rögnvaldsson, Einar Freyr Sigurðsson, and Joel Wallenberg. 2014. Rapid Deployment of Phrase Structure Parsing for Related languages: A Case Study of Insular Scandinavian. In *Proceedings of Ninth International Conference on Language Resources and Evaluation (LREC 2014)*, pages 91–95. European Language Resources Association (ELRA).

Svanhvít Lilja Ingólfsdóttir, Hrafn Loftsson, Jón Friðrik Daðason, and Kristín Bjarnadóttir. 2019. Nefnir: A high accuracy lemmatizer for Icelandic. In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pages 310–315, Turku, Finland. Linköping University Electronic Press.

Christo Kirov, Ryan Cotterell, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sebastian Mielke, Arya D McCarthy, Sandra Kübler, et al. 2018. Unimorph 2.0: universal morphology. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA).

Christo Kirov, John Sylak-Glassman, Roger Que, and David Yarowsky. 2016. Very-large scale parsing and normalization of wiktionary morphological paradigms. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 3121–3126. European Language Resources Association (ELRA).

Hrafn Loftsson. 2006. Tagging Icelandic text: An experiment with integrations and combinations of taggers. *Language Resources and Evaluation*, 40(2):175–181.

Arya D. McCarthy, Christo Kirov, Matteo Grella, Amrit Nidhi, Patrick Xia, Kyle Gorman, Ekaterina Vylomova, Sabrina J. Mielke, Garrett Nicolai, Miikka Silfverberg, Timofey Arkhangelskiy, Nataly Krizhanovsky, Andrew Krizhanovsky, Elena Klyachko, Alexey Sorokin, John Mansfield, Valts Ernštreits, Yuval Pinter, Cassandra L. Jacobs, Ryan Cotterell, Mans Hulden, and David Yarowsky. 2020. Unimorph 3.0: Universal morphology. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 3922–3931, Marseille, France. European Language Resources Association.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin-Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Ku-mar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *CoRR, abs/1701.03980*.

Anna Björk Nikulásdóttir, Jón Guðnason, and Steinþór Steingrímsson. 2017. *Mál tækni fyrir íslensku 2018–2022: verkáætlun [Language Technology for Icelandic 2018-2022: Strategic Plan]*. Mennta- og menningarmálaráðuneytið, Reykjavík, Iceland.

Jörgen Pind, Friðrik Magnússon, and Stefán Briem. 1991. *Íslensk orðtíðnibók [The Icelandic Frequency Dictionary]*. The Institute of Lexicography, University of Iceland, Reykjavík, Iceland.

Jóhan Hendrik W. Poulsen, Marjun Simonsen, Jógvan í Lón Jacobsen, Anfinnur Jóhansen, and Zakaris Svabo Hansen, editors. 1997. *Føroysk orðabók [Dictionary of Faroese]*. Føroya fróðskaparfelag, Torshavn.

Steinþór Steingrímsson, Örvar Kárason, and Hrafn Loftsson. 2019. Augmenting a BiLSTM Tagger with a Morphological Lexicon and a Lexical Category Identification Step. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 1161–1168, Varna, Bulgaria.

Trond Trosterud. 2009. A constraint grammar for Faroese. In *Proceedings of the 16th Nordic Conference of Computational Linguistics (NODALIDA 2009)*. Northern European Association for Language Technology (NEALT).

Höskuldur Þráinsson, Hjalmar P. Petersen, Jógvan í Lón Jacobsen, and Zakaris Svabo Hansen. 2004. *Faroese: An overview and reference grammar*. Føroya fróðskaparfelag, Torshavn.

Leveraging Multi-domain, Heterogeneous Data using Deep Multitask Learning for Hate Speech Detection

Prashant Kapil
Department of CSE
Indian Institute of Technology Patna
prashant.pcs17@iitp.ac.in

Asif Ekbal
Department of CSE
Indian Institute of Technology Patna
asif@iitp.ac.in

Abstract

With the exponential rise in user-generated web content on social media, the proliferation of abusive languages towards an individual or a group across the different sections of the internet is also rapidly increasing. It is very challenging for human moderators to identify the offensive contents and filter those out. Deep neural networks have shown promise with reasonable accuracy for hate speech detection and allied applications. However, the classifiers are heavily dependent on the size and quality of the training data. Such a high-quality large data set is not easy to obtain. Moreover, the existing data sets that have emerged in recent times are not created following the same annotation guidelines and are often concerned with different types and sub-types related to hate. To solve this data sparsity problem, and to obtain more global representative features, we propose a Convolution Neural Network (CNN) based multi-task learning models (MTLs)[1] to leverage information from multiple sources. Empirical analysis performed on three benchmark datasets shows the efficacy of the proposed approach with the significant improvement in accuracy and F-score to obtain state-of-the-art performance with respect to the existing systems.

1 Introduction

The continuing rise in the usage of the internet has loaded a large volume of content in social media like Twitter with 500 million[2] tweets per day and Facebook laden with 510K comments[3] per minute. These sites are important sources for people to give their opinion on a host of general social topics. The message can be clean, sarcastic, obscene, offensive, rude, hateful, etc. (Nockleby, 2000) defined *hate speech* as a broad umbrella term that describes it as any communication that demeans any person or any group based on race, color, gender, ethnicity, sexual orientation, and nationality.

Defining hate is, itself, a difficult task as it greatly depends on the demography, i.e. the same content comes under *Right to speech* in some countries, while other countries might adhere to a very strict policy for the same message.

In recent times, Germany made policy for the social media companies that they would have to face a penalty up to $60 million[4] if they failed to remove illegal content on time. Denmark and Canada have laws that prohibit all the speeches that contain insulting or abusive content that could promote violence and social disorders. The Indian government has also urged leading social media sites such as Facebook, Twitter to take necessary action against hate speech, especially those posts that create social outrage. Setting aside legal actions our aim should be to combat these texts by agreeing to a set of standard definitions, guidelines, and practices to remove the content. Recently many automated techniques following supervised learning utilizing deep neural networks have been developed. Recently shared tasks such as (Basile et al., 2019; Mandl et al., 2019; Zampieri et al., 2019) have mainly focused on developing multiple-layer identification of offensive languages. The existing prior research towards this direction mainly focused on single-task learning (STL) where classification task on one data set at a time is solved by training the model in stochastic gradient descent approach. However, training of neural networks relies on a large amount of data, and creating a balanced data set seems to be time-consuming, and tedious. As the number of posts showing aggressive tendencies is very less

[1] code is available at https://github.com/imprasshant/STL-MTL

[2] https://www.internetlivestats.com/twitter-statistics/

[3] https://kinsta.com/blog/facebook-statistics/

[4] https://www.inc.com/joseph-steinberg/germanys-tough-new-social-media-law-punishes-offensive-posts-with-fines-of-up-to-60-million.html

Proceedings of the 17th International Conference on Natural Language Processing, pages 491–500
Patna, India, December 18 - 21, 2020.

compared to non-aggressive posts, we leverage the concept of homogeneous multi-task learning where we utilized the multiple classification task data sets to be trained jointly to solve the task. Although binary classification is very problematic, it filters out harmful messages and provides hateful data to further train the model to classify the data into more fine-grained classes and helps in getting the target and sentiment behind the posts, thus preventing the violation of the right to freedom of speech.

The key characteristics of our current work are summarized as follows.
(i). We propose a deep multi-task learning framework that leverages information from multiple sources. We experiment on five different variations of CNN based single-task learning (STL) and five different variations of CNN based multi-task Learning (MTL) approaches for solving the problem of hate speech classification.
(ii). The proposed classification approach can be utilized to obtain hateful or abusive posts to further train any classifier with these data to perform the classification to finer labels.

2 Related Work

In recent times, online hate speech detection has attracted the attention of researchers and developers because of its necessity in maintaining social fabrics. In recent times, most of the methods that have emerged are mainly based on classical machine learning and deep learning. (Badjatiya et al., 2017) defined hateful tweets as speech that contains abusive speech targeting *individuals* (e.g. cyberbullying, politician, a celebrity or a product) or a *group* (a religious group, country, LGBT, gender, organization), etc. (Wulczyn et al., 2017) identified the personal attack as a binary classification problem and experimented with logistic regression and multilayer perceptron with word n-grams or char n-grams based features. (Nobata et al., 2016) observed that including simple n-gram features are more powerful than linguistics-based features, syntactic-based features, as well as word and paragraph embeddings. (Davidson et al., 2017) highlighted the differences between hateful and offensive languages and that, conflating these two erroneously will make many speakers be hate speakers. They highlighted the need to train the model with hateful data that does not contain any particular keywords or abusive terms to enrich the model with more contextual and knowledge-based features. (Chakrabarty et al., 2019) provided visualization of attention weights and concluded that the model assigned higher attention to potentially abusive terms when employed with contextual information in comparison to self-attention based features. (MacAvaney et al., 2019) utilized BERT, that make use of Transformer (Vaswani et al., 2017), an attention mechanism helping to capture the contextual representation between words and sub-words of a sentence that is utilized to perform the classification task on (de Gibert et al., 2018). (Pérez and Luque, 2019) leveraged BiLSTM with a dense layer on top consuming Elmo vectors (Peters et al., 2018), and Bag of words as additional input to do the classification on the data by (Basile et al., 2019). In this paper, we present a multi-task framework that aims at leveraging information contained in multiple related tasks and improve the classification performance of the hate data sets.

3 Methodology

3.1 Preprocessing

We perform different steps of pre-processing to clean the text.

1. A light pre-processing by removing all the characters like @!:;?. and removing all the numbers (0-9), URLs present in the tweet.

2. Word segmentation is being done to convert the hashtags like *#BuildTheWall* → build the wall, *#SendthemBack*→ send them back, *#refugeeswelcome* → refugees welcome, *#humantrafficking* → human trafficking, *#whitegenocides* → white genocides, *#makeLoveNotWar* → make love not war, *#F**kracism*→ f**k racism, etc. using python (Rossum, 1995) *word segment* to preserve the important features to compute sentiment of any type of message.

3. All the emoticons were manually categorized into 5 categories, i.e. *love*, *sad*, *happy*, *shocking* and *anger*. The unicode character of emoticons is then substituted with the token it matched.

4. All the @ mentions were replaced with the common token i.e *user*.

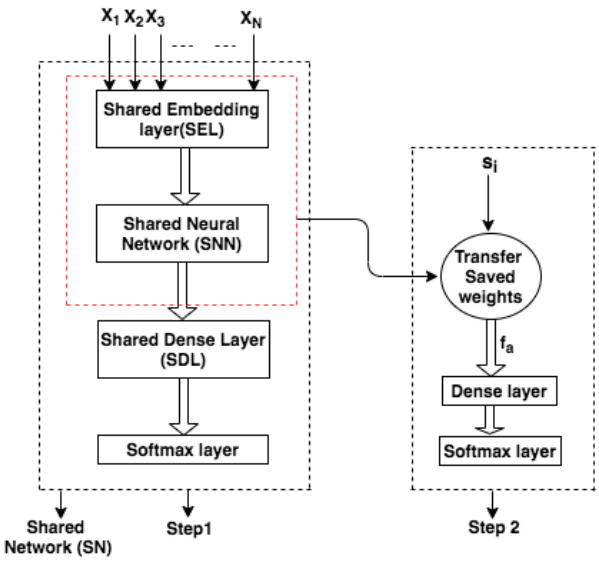


Figure 1: **Architecture of Fully Shared Multi-Task Learning(FS-MTL)**

3.2 Embeddings

Word embeddings (w_e): In our experiments, we utilized the Google pre-trained *word2vec* vectors trained on 100 billion words to produce 300 dimensions for each word capturing the semantic and syntactic relationship between the words trained using skip-gram by (Mikolov et al., 2013).
Character embedding (c_e): The presence of Out-of-Vocabulary (OOV) word is a serious problem in a social media text. Embedding for such words in the pre-trained word embedding model is not found, hence losing morphological information. We leverage the skip-gram model by (Bojanowski et al., 2017) which represents each word as a bag of character n-grams. The dimension of each word using character embedding is 300. The final word embedding x_e for word x $\in X$is represented by the following process:

$$x_e = w_e \oplus c_e \tag{1}$$

where ($\oplus$) denotes the concatenation operation and X is the number of unique tokens. The resulting dimension of x_e is 600.

3.3 Models

We adopt (Kim, 2014) for word-level Convolutional neural networks(*Word-CNN*) and (Zhang et al., 2015) for Character-level Convolutional neural networks (*Char-CNN*) to build different variants of CNN for our experiments. Here, we describe Word-CNN and Char-CNN.

Word-CNN: We adopted the CNN-Static by (Kim, 2014). The input sequence S_i of length *l* is tokenized to assign a unique integer index to each word w_i, that is then mapped to its *N* dimension real-valued vector. A *convolution* operation involves a filter $f \in R^{hN}$ which is applied to the *h* words to produce a new feature x_i in Eq.2. Here, *b* $\in R$ is a bias term and *g* is a non-linear activation function. This process is repeated *l-h+1* to get the feature map *x* in Eq.3.

$$x_i = g(f \cdot S_{i:i+h-1} + b) \tag{2}$$

$$x = [x_1, x_2, ..., x_{l-h+1}] \tag{3}$$

Then the pooling layer is applied to reduce the spatial size of the representation helping in reducing over-fitting. The vector form of features obtained from the last CNN layer is fed into the fully connected layer followed by the softmax activation function that calculates the probability values for all the classes. We define the following 5 models that utilize the word-level CNN: *Model 1*[5], *Model 2*, *Model 4*, *Model 5*, *Model 6 Model 7*, *Model 8*, *Model 9* and *Model 10*(all these models are defined below).

Char-CNN: We adopt the character-CNN (Zhang et al., 2015) where the gradients are obtained by backpropagation to perform optimization. It accepts a sequence of encoded characters as input. The encoding is done by quantizing each

[5]Model *i* and model *i* will refer to the same model in the text where $i \in [1, 10]$

character using 1-of-*m* encoding, also known as *one-hot-encoding*. Then the sequence of characters is transformed to a sequence of *m* sized vector with fixed length l_o= 256/1024. The value of *m* in their proposed model is 70 with 26 for the English alphabet, 10 digits, 33 other characters, and one for the newline character. They designed 9 layers with 6 convolutional layers and 3 fully-connected layers. They initialized the weights using Gaussian distribution. Two models, *viz. Model 3* and *Model 4* (all these models are defined below) utilize the concept of character CNN. Below, we briefly describe each of the proposed models.

Model 1: *Random word vectors-CNN*: We adopt the method by (Gambäck and Sikdar, 2017) to assign the random vector of dimension 600 as feature embeddings for words.

Model 2: *Word-CNN*: In this model, we utilize real-valued vectors of 600 dimensions for each word capturing the semantic and syntactic relationship between the words from Eq.1.

Model 3: *Char-CNN*: Our designed model consists of representing each characters using 27 sized vector with 26 elements for the English alphabet and one for all other symbols. This model consists of a convolution layer with kernel size 4 followed by max-pool layer of size 3. This is fed into another convolution layer with kernel size 4 and max-pool layer of size 3. This is followed by 2 dense layers of size 64 and 2. The strides used in convolution layers are 4 and 2.

Model 4: *Hybrid-CNN*: It utilizes both character and word input at the same time. The output of both the channels after flattening the pooling features is concatenated to pass into a fully connected layer with softmax activation function.

Model 5: *CNN-Word-Attention:* This mechanism expands the functionality of neural networks by paying attention to the specific parts of the sentence depicting the human brain. We utilized the CNN-sentence-level attention by (Raffel and Ellis, 2015). It calculates the attention weight for the important words to form a representation of the sentences. Each word's hidden state representation (h_t) is passed through a learnable function a(h_t) to produce probability value α_1,α_1...α_n for each word. The sentence vector *output* is calculated by the weighted average of h_t with weights of α.

$$e_t = tanh(Wh_t + b) \tag{4}$$

$$\alpha_t = softmax(e_t) \tag{5}$$

$$output = \sum_{t=1}^{t=n} \alpha_t h_t \tag{6}$$

Fully Shared MTL(FS-MTL): The architecture of model 6, model 7 and model 8 are based on this scheme that is shown in Figure 1. This scheme consists of two steps.

Step 1: Training of Shared Network (SN): The SN consists of 4 components: Shared Embedding Layer (SEL), Shared Neural Network (SNN), Shared Dense Layer (SDL) and Softmax layer. This network is pre-trained by taking equal samples from each of the participating data sets and training it in batch-wise manner. The Shared Embedding Layer (SEL) consists of the unique tokens from all the data sets. All the different classes of each data set are merged to represent class c_i where in this experiment i $\in$ [1,N] where N is number of data set. The parameters of the SN are trained to minimize the *categorical cross entropy* of the predicted and true distribution on all the tasks. The loss L_{Task} can be defined as:

$$L_{Task} = \sum_{k=1}^{K} \alpha_k \cdot L(\hat{y}^k, y^k) \tag{7}$$

where α_k is the class weight i.e 1 in this experiment and L($\hat{y}$,y) is defined in equation 8.

$$L(\hat{y}, y) = -\sum_{i=1}^{C}\sum_{j=1}^{N} y_i^j \log \hat{y}_i^j \tag{8}$$

Here C is the total number of classes, N is the number of samples, y_i^j is the ground truth label and $\hat{y}_i^j$ is the predicted label.

Step 2: The trained shared network (SN) is sliced off to extract the weight matrix of the first two layers: SEL and SNN, denoted in red color in Figure 1[6]. The parameters of the transferred layers to the new network are kept frozen. A new sentence S_i is passed through the frozen weight matrix to get representation f_a which is passed to dense layer followed by softmax layer to get the probability values.

Model 6: *Word-CNN-Fully Shared MTL*: We adopt the schema of (Liu et al., 2017) by

[6]The figure is best viewed in color

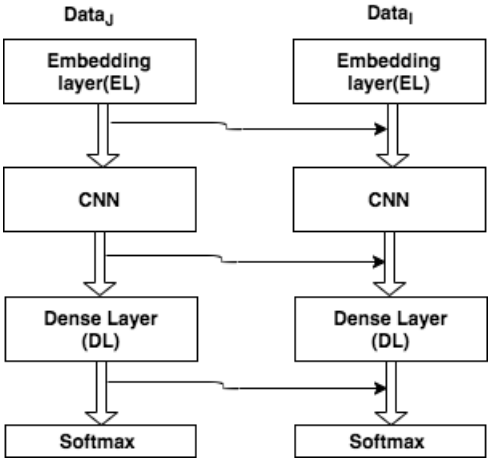


Figure 2: **Architecture of Soft Sharing MTL**

employing fully-shared Word-CNN layers to extract features for all the tasks. It takes the view that features of *task m* can be totally shared by *task n* and the vice-versa. Figure 1 explains the idea. Here $X_1 \rightarrow D_1$, $X_2 \rightarrow D_2$, $X_3 \rightarrow D_3$.{D_1, D_2 and D_3 explained in section 4}

Model 7: This model utilizes the sentiment data to be trained with hate data in a fully shared manner. Here $X_1 \rightarrow D_i$, $X_2 \rightarrow S_1$, $X_3 \rightarrow S_2$, $X_4 \rightarrow S_3$.
{S_1, S_2 and S_3 explained in section 4} and i = 1,2,3

Model 8: The intermediate feature f_a obtained from model 6 and model 7 is concatenated to pass into dense layer followed by softmax layer.

Model 9: *Soft Sharing CNN-Word-MTL:* This model is motivated by (Xiao et al., 2018) that utilizes the CNN based multitasking paradigm. Every task owns a subnet and shares the features with each other. The embedding layer (EL) in Figure 2 consists of uniques tokens present in all the data sets. Here D_1, D_2 and D_3 will share feature with each other. All the subnet undergoes a pre-training of the text sequences. Let C be the total collection of *n* tasks C = {T_1,T_2.....T_n}. The output of any sequence s_i at any layer *l* is the concatenation of the output of the same sequence s_i from all the other tasks. Task *i* borrows the features from Task *j* which is calculated as

$$g_{ij}^{l} = (W_{ij}^{l} \cdot F_{ij}^{l} + b_{ij}^{l}) \tag{9}$$

where l denotes the level of layers. For any task i the output of F_{ij}^{l+1} by merging the F^l from all the other tasks by

$$F_{ij}^{l+1} = \sum_{j \in Cj \neq i} g_{ij}^{l} + F_{i}^{l} \tag{10}$$

Model 10: The training will remain the same as of model 9 but D_i will share the features with S_1, S_2 and S_3. Figure 2 explains the idea for 2 task which can be extended to *n* tasks. Here i $\in$ [1,3]

4 Dataset and Experiment Setup

4.1 Data

We evaluate our model on 3 different datasets. (denoted as D_1, D_2 and D_3). Hate speech and sentiment analysis are closely related, and it is safe to assume that usually negative sentiment pertains to a hate speech message. We also utilized 3 sentiment data which have been described as S_1, S_2 and S_3. Table 1 and Table 2 shows the statistics of the datsets.

D_1(de Gibert et al., 2018): The sentences have been extracted from *stormfront*, a white supremacist forum. A subset of 22 sub-forums covering diverse topics and nationalities was random-sampled to gather individual posts uniformly distributed among the sub-forums and users. The most common hateful words found were *ape*,*homosexuals*, *libtard*, *monkey* and *miglet*. The data set constitutes of 36.05% and 41.63% hate vocabulary from *gender* and *ethnicity*.

D_2(Basile et al., 2019): This dataset is part of hate speech against immigrants and women in English, collected between July to September 2018. The most frequent keywords were *migrant*,

refugee, *b**ch*, *#buildthatwall*, *h*e* and *women*.

D_3(Mandl et al., 2019): The HASOC dataset was subsequently sampled from Twitter and partially from Facebook for the three languages. For our experiments, we leveraged the English data. They identified topics for which many hate posts can be expected. Thus, the tweets were acquired using hashtags and keywords that contained offensive contents.

S_1[7]: This dataset is crawled from twitter containing US Airline Sentiment tweets.

S_2(Rosenthal et al., 2017): The English topics based on popular current events that were trending on Twitter, both internationally and in specific English speaking countries were used to crawl using Twitter API. The topics included a range of named entities (e.g., Donald Trump, iPhone), geopolitical entities (e.g., Aleppo, Palestine), and other entities (e.g., Syrian refugees, Dakota Access Pipeline, Western media, gun control, and vegetarianism).

S_3(Misra and Arora, 2019): To overcome the limitations related to noise in Twitter datasets, they collected a news Headlines dataset from two news websites. The Onion2 aims at producing sarcastic versions of current events and they collected all the headlines from News in Brief and News in Photos categories (which are sarcastic). They collect real (and non-sarcastic) news headlines from HuffPost.

Table 1: **Hateful Data Statistics**

Dataset	labels and count	Test
D_1	Hate:1097 Non-Hate:8571	CV
D_2	Hate:4210 Non-Hate:5790	Hate:1260 Non-Hate:1740
D_3	Hate:2261 Non-Hate:3591	Hate:288 Non-Hate:865

4.2 Experimental Setup

All the deep learning models were implemented using Keras, a neural network API by (Chollet et al., 2018) with Tensorflow (Abadi et al., 2016) at the backend. All the dataset were split using 5-fold cross validation mode using the Stratified K-fold with 80% for training and 20% for testing with an equal proportion of samples from all the classes. The batch size of 30 is used for training SN Model in Figure 1. The official test set of 3000 instances is utilized for D_2 and 1153 for D_3. Random search was done to fine-tune the results of the neural networks to select the best performing combination of hyperparameters. Categorical cross-entropy is used as loss function with *adam*, a combination of *Adagrad* and *RmsProp* is used as the optimizer. The number of filters used for Word-CNN and Char-CNN are 100 and 256. The value for *bias* is randomly initialized to all zeros, *relu* activation function were employed at the intermediate layer and *softmax* is utilized at last dense layer.

Table 2: **Sentiment Data Statistics**

Dataset	labels and count	Test
S_1	Positive:2363 Negative:9178 Neutral:3099	-
S_2	Positive:7059 Negative:3231 Neutral:10342	-
S_3	Sarcastic:25358 Non-sarcastic:29970	-

5 Results, Comparison and Analysis

We report the 5-Fold cross validation result for D_1, D_2 and D_3 in Table 5, Table 6 and Table 7. Of the 10 models all the 5 CNN-MTL[8] outperforms 5 CNN-STL based approaches. We report here the macro-f, weighted-f and accuracy of the proposed methods. The best models for STL in D_1, D_2 and D_3 are CNN-attention, character-CNN and CNN-attention respectively. Hybrid-CNN also shows significant improvements in D_2. For the MTL based approach, concatenation of sentiment and hate based features gives good results for D_1. For D_2 and D_3 model 7 and model 6 performs best. Table 8, Table 9 and Table 10 enlist the comparisons between the previous benchmarks with our best models.

[7]https://www.kaggle.com/crowdflower/twitter-airline-sentiment

[8]The results obtained for S_1, S_2 and S_3 in MTL setting is not reported to focus only on hateful text detection.

Table 3: **False Negatives**

Sl	Type	Sentence
1.	Toxic	B**ch our streak is dying @C**t***Lady
2	Toxic	@AMike4761 Wake the f**k up and fight back! Savethewest sendthemback
3	Non-Toxic	Fed up with this crap! #DeportThemAll
4	Non-Toxic	@globalnews #SendThemHome we do not need those #Students here in #Canada
5	Direct Attack	”The first sexual attack against a woman happened in Hungary. The criminal is a ””legal”” afghan rapefugee. This is how Orban protects us.”
6	Direct Attack	@whaas3 @judithineuropa Just got on twitter because of this farce today. Imagine this, I make a report on You. Calling you names and telling people how big liar and a****le you are without reason. Would you be angry
7	Doubtful	@DVATW @TheHairyJobbie By god there is a lot of woman and children got off that boat eh,poor holiday makers paying good money to go on holiday to witness that roundthemup sendthemback
8	Doubtful	@AdamBandt if its gets you upset - Hes the best man for the job. #gohome #strongborders #sendthemback
9	Sarcastic	”Please don’t call it ””rescue”” - it’s human trafficking #portsclosed #sendthemback #benefitseekers”

Table 4: **False Positives**

Sl	Type	Sentence
1	Toxic	H*e stood behind a car door and said ”I don’t feel comfortably with you that close” b**ch made
2.	Toxic	@Cornjdw Lmao f**k you bitch don’t get mad at me cuz u don’t know the game of basketball hoe
3.	Non-Toxic	A little louder @w_terrence for the liberals in the back. #SendThemBack #BuildTheWall
4	Non-Toxic	Meanwhile in Spain..stopimmigration
5	Direct Attack	@realDonaldTrump Do you support @realDonaldTrump’s Southern Border Wall? Vote #RETWEETCheck out wall progress at Order Bricks to show you.
6	Direct Attack	Still can’t be. Even the neonazis behind the ””rapefugee”” website only claim around 450 rapes by immigrants for all of 2016.
7.	References	What an idiot. #buildthatwall
8.	References	@RepLowenthal Asylum seekers should enter at a LEGAL #USA port of entry

Table 5: **Evaluation Results on D_1 (de Gibert et al., 2018)**

Model	Macro(%)	Weighted(%)	Acc.(%)
Single Task Learning			
Model-1	47.44	83.30	88.39
Model-2	47.25	83.36	88.61
Model-3	**48.48**	83.13	87.47
Model-4	47.07	83.33	88.63
Model-5	47.79	**83.49**	**88.65**
Multi Task Learning			
Model-6	87.81	95.13	95.18
Model-7	85.19	93.93	93.84
Model-8	**90.55**	**96.35**	**96.52**
Model-9	84.17	93.70	93.78
Model-10	72.11	89.76	90.81

Table 6: **Evaluation Results on D_2 (Basile et al., 2019)**

Model	Macro(%)	Weighted(%)	Acc.(%)
Single Task Learning			
Model-1	46.69	44.33	50.76
Model-2	48.04	45.91	51.36
Model-3	**51.49**	**51.19**	51.56
Model-4	49.78	47.66	**52.76**
Model-5	45.47	42.76	50
Multi Task Learning			
Model-6	91.40	91.57	91.54
Model-7	**93.60**	**93.75**	**93.75**
Model-8	93.41	93.56	93.56
Model-9	90.11	90.35	90.35
Model-10	89.43	89.72	89.76

5.1 Error Analysis

Quantitative Analysis: The confusion matrix obtained by best performing model on D_1, D_2 and D_3 is presented in Table 11, Table 12 and Table 13. For D_1 model 8 performs best, model 7 is performing best for D_2 and D_3. From the table it can be seen that misclassification rate in the proposed model for *hate* is 23% for D_1, 31.5% for D_2 and 34% for D_3. However, the misclassification for *Non-Hate* is 1.6% for D_1, 54.25% for D_2 and 13.87% for D_3.

Qualitative Analysis: We also identified some of the false negative cases i.e hateful tweet predicted to non-hate and false positive cases i.e non-hate tweet classified as hateful class in Table 3

Table 7: **Evaluation Results on D_3 (Mandl et al., 2019)**

Model	Macro(%)	Weighted(%)	Acc.(%)
Single Task Learning			
Model-1	57.32	61.47	65.12
Model-2	56.98	61.08	64.54
Model-3	44.27	51.37	61.77
Model-4	59.82	63.14	65.12
Model-5	**62.19**	**65.27**	**67.05**
Multi Task Learning			
Model-6	**87.37**	**87.99**	**87.96**
Model-7	79.82	81.20	81.65
Model-8	86.76	87.65	87.93
Model-9	86.17	86.94	87.01
Model-10	84.34	85.13	85.12

Table 8: **Comparison to the state-of-the-art systems and proposed system for D_1 (de Gibert et al., 2018)**

Model	Macro(%)	Acc(%)
(MacAvaney et al., 2019)	82.01	82.01
(MacAvaney et al., 2019)	80.31	80.33
(Berglind et al., 2019)	70.80	72.20
(Berglind et al., 2019)	81.90	81.60
(Berglind et al., 2019)	78.40	77.10
Model-8	**90.55**	**96.52**
Model-6	87.81	95.18

Table 9: **Comparison to the state-of-the-art systems and proposed system for D_2 (Basile et al., 2019)**

Model	**Test Data**	
	Macro(%)	Acc(%)
(Ding et al., 2019)	54.60	**56**
(Montejo-Ráez et al., 2019)	51.90	-
(Pérez and Luque, 2019)	47.10	50.80
(Baruah et al., 2019a)	51	54
Model-7	**55.24**	55.26
Model-8	50.55	52.60

Table 10: **Comparison to the state-of-the-art systems and proposed system for D_3 (Mandl et al., 2019)**

Model	**Test Data**	
	Macro(%)	Acc(%)
(Mishra and Mishra, 2019)	74.65	-
(Baruah et al., 2019b)	74.62	-
(Jiang, 2019)	74.31	-
Model-7	**75.39**	**81.09**
Model-8	74.68	80.65

Table 11: **Confusion matrix of D_1 (de Gibert et al., 2018)**

Class	Hate	Non-Hate
Hate	844	253
Non-Hate	136	8435

and Table 4. It points to the fact that due to usage of words like f**k, bi**h, kill, bas***d etc. in both hate and non-hate context, the neural network is being confused to classify correctly.

Table 12: **Confusion matrix of D_2 (Basile et al., 2019)**

Class	Hate	Non-Hate
Hate	862	398
Non-Hate	944	796

Table 13: **Confusion matrix of D_3 (Mandl et al., 2019)**

Class	Hate	Non-Hate
Hate	190	98
Non-Hate	120	745

6 Conclusion and Future work

In this paper we have proposed five multi-task learning based approaches for hate speech detection. The proposed approaches has an ability to learn shared features between three different hate speech data sets and also leveraging the knowledge from the data of sentiment analysis tasks. The efficacy of the proposed approach is evident from the fact that it shows a consistent improvement in the F-score and accuracy values over the models working on single-task learning paradigm.

The system failure on some cases highlights the need to build a more diverse and robust neural network system to take into account the contextual, demographic as well as the knowledge based features.

Acknowledgement

The Authors gratefully acknowledge the project "HELIOS - Hate, Hyperpartisan, and Hyperpluralism Elicitation and Observer System", sponsored by Wipro Ltd.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760.

Arup Baruah, Ferdous Barbhuiya, and Kuntal Dey. 2019a. Abaruah at semeval-2019 task 5: Bi-directional lstm for hate speech detection. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 371–376.

Arup Baruah, Ferdous Ahmed Barbhuiya, and Kuntal Dey. 2019b. Iiitg-adbu at hasoc 2019: Automated hate speech and offensive content detection in english and code-mixed hindi text. In *FIRE (Working Notes)*, pages 229–236.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Nozza Debora, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, Manuela Sanguinetti, et al. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *13th International Workshop on Semantic Evaluation*, pages 54–63. Association for Computational Linguistics.

Tor Berglind, Björn Pelzer, and Lisa Kaati. 2019. Levels of hate in online environments. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 842–847.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Tuhin Chakrabarty, Kilol Gupta, and Smaranda Muresan. 2019. Pay "attention" to your context when classifying abusive language. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 70–79.

François Chollet et al. 2018. Keras: The python deep learning library. *ascl*, pages ascl–1806.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. *arXiv preprint arXiv:1703.04009*.

Yunxia Ding, Xiaobing Zhou, and Xuejie Zhang. 2019. Ynu_dyx at semeval-2019 task 5: A stacked bigru model based on capsule network in detection of hate. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 535–539.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using convolutional neural networks to classify hate-speech. In *Proceedings of the first workshop on abusive language online*, pages 85–90.

Ona de Gibert, Naiara Perez, Aitor García-Pablos, and Montse Cuadros. 2018. Hate speech dataset from a white supremacy forum. *arXiv preprint arXiv:1809.04444*.

Aiqi Jiang. 2019. Qmul-nlp at hasoc 2019: Offensive content detection and classification in social media. In *FIRE (Working Notes)*, pages 254–262.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742*.

Sean MacAvaney, Hao-Ren Yao, Eugene Yang, Katina Russell, Nazli Goharian, and Ophir Frieder. 2019. Hate speech detection: Challenges and solutions. *PloS one*, 14(8):e0221152.

Thomas Mandl, Sandip Modha, Prasenjit Majumder, Daksh Patel, Mohana Dave, Chintak Mandlia, and Aditya Patel. 2019. Overview of the hasoc track at fire 2019: Hate speech and offensive content identification in indo-european languages. In *Proceedings of the 11th Forum for Information Retrieval Evaluation*, pages 14–17.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Shubhanshu Mishra and Sudhanshu Mishra. 2019. 3idiots at hasoc 2019: Fine-tuning transformer neural networks for hate speech identification in indo-european languages. In *FIRE (Working Notes)*, pages 208–213.

Rishabh Misra and Prahal Arora. 2019. Sarcasm detection using hybrid neural network. *arXiv preprint arXiv:1908.07414*.

Arturo Montejo-Ráez, Salud María Jiménez-Zafra, Miguel A García-Cumbreras, and Manuel Carlos Díaz-Galiano. 2019. Sinai-dl at semeval-2019 task 5: Recurrent networks and data augmentation by paraphrasing. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 480–483.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th international conference on world wide web*, pages 145–153.

JT Nockleby. 2000. 'hate speech in encyclopedia of the american constitution.

Juan Manuel Pérez and Franco M Luque. 2019. Atalaya at semeval 2019 task 5: Robust embeddings for tweet classification. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 64–69.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Colin Raffel and Daniel PW Ellis. 2015. Feed-forward networks with attention can solve some long-term memory problems. *arXiv preprint arXiv:1512.08756*.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th international workshop on semantic evaluation (SemEval-2017)*, pages 502–518.

Guido Rossum. 1995. Python reference manual.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1391–1399.

Liqiang Xiao, Honglun Zhang, and Wenqing Chen. 2018. Gated multi-task network for text classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 726–731.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). *arXiv preprint arXiv:1903.08983*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Author Index