ICON 2021

18th International Conference on Natural Language Processing

Proceedings of the Conference

December 16 - 19, 2021
National Institute of Technology Silchar, India

©2021 NLP Association of India (NLPAI)

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-2021. 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 204 submissions, 78 papers were selected, 51 long papers, 27 short papers, 2 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 8 tutorials, 4 workshop, 2 shared tasks, and 3 system demonstrations.

We are deeply grateful to Prof. Josef van Genabith from DFKI and Saarland University (Germany), Prof. Philip Resnik from University of Maryland (USA), Prof. Rada Mihalcea from University of Michigan (USA) and Dr. Louis-Philippe Morency from Carnegie Mellon University (USA) for giving the keynote lectures at ICON-2021.

We thank all the area chairs for the various tracks at ICON-2021, especially, Dipti Misra Sharma, Pawan Goyal (Syntax and Semantics), Kamal Kumar Choudhary, Krishna Prasad Miyapuram (Computational Psycholinguistics), Asif Ekbal, Shubhashis Sengupta (QA, Information Extraction), Ranjani Parthasarathi, Girish K. Palshikar (Information Retrieval and Text Mining), Dipankar Das, Raksha Sharma (Sentiment Analysis and Emotion Recognition), Girish Nath Jha, Niladri Shekhar Dash (Language Resources and Evaluation), Srinivas Bangalore, Kalika Bali (Speech), C V Jawahar, Asutosh Modi (Multimodality), Anoop Kunchukuttan, Karunesh Arora (Machine Translation), Monojit Choudhury, Anand Kumar M (NLP Applications), Sudeshna Sarkar, Sandipan Dandapat (Machine Learning in NLP), Vasudeva Varma, Sriparna Saha (Natural Language Text Generation). We also thank Radhika Mamidi, Samar Husain (Doctoral Consortium), Partha Pakray, Vishal Goyal (Shared Task/Tools Contest), Sudip Kumar Naskar, Amitava Das (Workshop/Tutorial) for taking the responsibilities of the events.

We are thankful to the team members of the Centre for Natural Language Processing (CNLP) Group and the Department of Computer Science and Engineering for making the organization of the event at the National Institute of Technology Silchar (NIT Silchar) a success.

We heartily express our gratitude to Partha Pakray, Naresh Babu Muppalaneni, Badal Soni, Loitongbam Sanayai Meetei, Ringki Das, Salam Michael Singh, Alok Singh for their timely help with sincere dedication to make this conference a success. We also thank all the student volunteers 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-2021, 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 2021
Silchar

Sivaji Bandyopadhyay - PC Co-chair
Sobha L - PC Co-chair
Pushpak Bhattacharyya - General Chair

Thoudam Doren Singh - Organizing Committee Chair

Advisory Committee Chair

Rajeev Sangal, IIIT Hyderabad, India

Conference General Chair

Pushpak Bhattacharyya, IIT Bombay, India

Program Chairs:

Sivaji Bandyopadhyay, NIT Silchar, India (Co-Chair)
Sobha L, AU-KBC Research Centre, India (Co-Chair)

Organizing Chair:

Thoudam Doren Singh, NIT Silchar, India

Program Committee:

Dipti Misra Sharma, IIIT Hyderabad
Pawan Goyal, IIT Kharagpur
Kamal Kumar Choudhary, IIT Ropar
Krishna Prasad Miyapuram, IIT Gandhinagar
Asif Ekbal, IIT Patna
Shubhashis Sengupta, Accenture
Ranjani Parthasarathi, Anna University
Girish K. Palshikar, TCS Innovation
Dipankar Das, Jadavpur University
Raksha Sharma, IIT Roorkee
Girish Nath Jha, JNU
Niladri Shekhar Dash, ISI Kolkata
Srinivas Bangalore, Interactions LLC, USA
Kalika Bali, Microsoft Research India
C V Jawahar, IIIT Hyderabad
Asutosh Modi, IIT Kanpur
Anoop Kunchukuttan, Microsoft
Karunesh Arora, CDAC Noida
Monojit Choudhury, Microsoft
Anand Kumar M, NIT Surathkal
Sudeshna Sarkar, IIT Kharagpur
Sandipan Dandapat, Microsoft
Vasudeva Varma, IIIT Hyderabad
Sriparna Saha, IIT Patna

Doctoral Consortium Chairs

Radhika Mamidi, IIIT Hyderabad
Samar Husain, IIT Delhi

Shared Task/Tools Contest Chairs

Partha Pakray, NIT Silchar
Vishal Goyal, Punjabi University, India

Workshop/Tutorial Chairs

Sudip Kumar Naskar, Jadavpur University
Amitava Das, Wipro AI

Invited Speakers

Prof. Josef van Genabith, DFKI and Saarland University, Germany
Prof. Philip Resnik, University of Maryland, USA
Prof. Rada Mihalcea, University of Michigan, USA
Dr. Louis-Philippe Morency, Carnegie Mellon University,USA

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.

Abhay Garg, TCS Research Pune, India
Abhijith Athreya Mysore Gopinath, Pennsylvania State University, USA
Abir Chakraborty, Microsoft India
Abu Kaisar Mohammad Masum, Daffodil International University
Aditya Joshi, CSIRO, India
Aditya Mogadala, Saarland University, Germany
Aditya Shah, IIT Indore, India
Aditya Srivastava, IIIT Hydebad, India
Adwait Ratnaparkhi, University of California Santa Cruz, USA
Adway Mitra, IIT Kharagpur
Aiusha Hujon, NEHU, Shillong
Akshay Parekh, IITGuwahati
Àlex Atrio, HEIG-VD and EPFL, Switzerland
Ali Saeedi, Kent State University
Alok Singh, NIT Silchar
Amba Kulkarni, University of Hyderabad, India
Amit Kumar, Université de Caen
Amitava Das, Wipro AI Lab, Bangalore, India
Amrit Verma, Siemens, India
Anand Kumar M, NIT Surathkal
Andrei Popescu-Belis, HEIG-VD and EPFL, Switzerland
Angel De Paula Magnossão, PRHLT, Spain
Anmol Nayak, ARiSE Labs, Bosch
Anupam Jamatia, NIT Agartala
Anupam Mondal, Jadavpur University, India
Anurag Dwarakanath, Amazon, Bangalore, India
Anustup Bhattacharya,
Apoorva Singh, IIT Patna
Arnab Bhattacharya, IIT Kanpur
Ashis Chanda, Temple University, USA
Ashraf Kamal, ACL Digital, India
Asif Ekbal, IIT Patna
Atul Kumar, Roku, USA
Ayan Das, BITS Pilani, Hyderbad
Badal Soni, NIT Silchar
Bharatram Natarajan, SRIB, India
Bibekananda Kundu, CDAC,Kolkata
Bishwa Das, Maharaja Sriram Chandra Bhanja Deo University
Björn Gambäck, Norwegian University of Science and Technology; RISE, Sweden
Bo Dong, University of Texas at Dallas
Braja Gopal Patra, Weill Cornell Medicine, New York, USA
Bratati Dhar, ISI Kolkata
Brijesh Bhatt, Dharmsinh Desai University, Nadiad
Carlos Escolano, Universitat Politècnica de Catalunya, Barcelona
Chandni Saxena, The Chinese University of Hong Kong, Shatin, NT, Hong Kong
Chandrahas, IIS Bengaluru, India
Chandresh Maurya, IIT Indore

Chayanika Chatterjee,
Chenchen Ding, National Institute of Information and Communications Technology, Japan
Chiranjeevi Yarra, IIIT Hyderabad
Cristina España-Bonet, DFKI and Saarland University
Danish Ebadulla, PES University Bangalore, India
Deeksha Varshney, IIT Patna
Deepali Jain, NIT Silchar
Dipankar Das, Jadavpur University, Kolkata
Diptesh Kanojia, University of Surrey, United Kingdom
Ganesh Mirishkar, IIIT,Hyderabad
Gatha Sharma, Shiv Nadar University
Gihan Dias, University of Moratuwa, Sri Lanka
Girish Jha, Jawaharlal Nehru University, India
Gouri Sankar Majumder, IIT Guwahati
Gunjan Ansari, JSS ATE-Noida
Hanumant Redkar, IIT Bombay, India
Himangshu Sarma, IIIT Sri City, Chittoor
Hour Kaing, National Institute of Information and Communications Technology, Japan
Hridoy Sankar Dutta, University of Cambridge
Indranil Roy, Jadavpur University
Jingxuan Yang, Beijing University
Joel Beckles, University of the West Indies
Josef van Genabith, DFKI and Saarland University
Juan Javier Sánchez Junquera, Universitat Politècnica de València
Jyotisankar Kalia, Gandhi Institute for Education and Technology (GIET), Khurda, Bhubaneswar
K. Sreenivasa Rao, IIT Kharagpur
Kamal Sarkar, Jadavpur University
Kanika Kalra, TCS Research Pune
Katsuhito Sudoh, Nara Institute of Science and Technology, Japan
Kaustubh Agarwal, Netaji Subhas University of Technology New Delhi
Kengatharaiyer Sarveswaran, University of Jaffna, Sri Lanka
Komal Bharti, IIT Guwahati
Loitongbam Sanayai Meetei, NIT Silchar
Lov Kumar, BITS Pilani Hyderabad
Lukas Edman, University of Groningen
Malhar Kulkarni, IIT Bombay
Manish Verma, IBM India
Mauajama Firdaus, IIT Patna
Mayur Patidar, TCS Research Pune
Meghna Majumder
Michael Carl, Kent State University, USA
Minghan Wang, Huawei
Minh Thuan Nguyen, University of Engineering and Technology, VNU Hanoi
Mithun Kumar S R, BITS Pilani
Mohd Fazil, MITS Madanapalle
Mohd Zeeshan Ansari, Jamila Mia Islamia
Monalisa Dey, Jadavpur University
Mukesh Kumar Rohil, BITS Pilani
Musica Supriya, Manipal Institute of Technology, India
Muskan Garg, Thapar Institute of Engineering and Technology, India
Ngo Xuan Bach, Posts and Telecommunications Institute of Technology (PTIT), Hanoi, Vietnam

Nidhi Arora, Interactions
Niladri Dash, ISI Kolkata
Niraj Kumar, Samsung
Nirnoy Dutta
Nitin Ramrakhiyani, Tata Research Development and Design Centre (TRDDC) in Pune
Niyati Bafna, Faculty of Mathematics and Physics, Charles University
Paolo Rosso, Universitat Politècnica de València
Parameswari Krishnamurthy, University of Hyderabad
Partha Pakray, NIT Silchar
Peter Scharf, Indian Institute of Advanced Study, Shimla
Pracheta Sahoo, University of Texas at Dallas
Pradeep Kumar Roy, IIIT Surat
Prajit Dhar, University of Groningen
Prajna Upadhyay, Research Engineer, CEDAR Team, INRIA, France
Pramit Bhattacharyya, IIT Kanpur
Pranav Goel, Wadhwani AI
Pritam Deka, Queen's University Belfast, UK
Priyatam Naravajhula, CBIT, Hyderabad
Punnoose A K, Flare Speech Systems
Rabul Laskar, NIT Silchar, India
Radha Krishna Guntur, VNRVJIET, Hyderabad
Rajaswa Patil, TCS Research - Tata Consultancy Services Limited
Rajat Kumar, TCS Research
Rajendra Roul, Thapar Institute of Engineering and Technology
Raksha Sharma, IIT Roorkee
Ramakrishna Appicharla, Indian Institute of Technology Patna
Ramesh Dadi, SR University, India
Ramya Akula, University of Central Florida
Ranjani Parthasarathi, Anna University
Ratnakar Pawar, Sony Playstation ML / Illinois Tech Chicago
Ravi Shekhar, Queen Mary University of London
Redwan Rizvee, Department of Computer Science and Engineering, University of Dhaka
Reynier Ortega Bueno, PRHLT Research Center, UPV
Riccardo Cervero, Università degli Studi di Milano Bicocca, Italy
Rina Kumari, IIT Patna
Ringki Das, NIT Silchar
Ritesh Kumar, Dr. Bhimrao Ambedkar University
Rusa Bhowmik, Jadavpur University
Sachin Pawar, TCS Research, Pune
Sadat Shahriar, University of Houston
Sai Muralidhar Jayanthi, MindMeld, Cisco Systems Inc.
Sainik Mahata, Institute of Engineering and Management
Salam Michael Singh, NIT Silchar
Samah Alazani, Dr. Babasaheb Ambedkar Marathawada University, Aurangabad, India
Samir Karmakar, Jadavpur University
Samudra Vijaya, IIT Guwahati
Sandhya Singh, CFILT, IIT Bombay
Sandip Sarkar, Jadavpur University
Sangameshwar Patil, TCS and Doctoral Research Scholar, IIT Madras
Sangram Kapre, Playstation
Sanjukta Ghosh, IIT BHU

Santanu Pal, Wipro AI
Santosh Kumar Mishra, Indian Institute of Technology Patna
Sara Renjit, Cochin University of Science and Technology, India
Satoshi Nakamura, Nara Institute of Science and Technology, Japan
Saujas Vaduguru, IIIT Hyderabad
Saumajit Saha, TCS Research
Seba Susan, Delhi Technological University
Shaily Bhatt, Predoctoral Researcher, Google Research
Shantipriya Parida, Silo AI
Shubhashis Sengupta, Accenture Labs
Siddhartha Mukherjee, SRIB India
Sixing Lu, Amazon Alexa AI
Soma Paul, IIIT-Hyderabad
Soumik Mandal, Ohio State University
Soumitra Ghosh, IIT Patna
Soumyadeep Ghosh, Mastercard
Sovan Kumar Sahoo, Indian Institute of Technology Patna
Srija Deb, ISI Kolkata
Sripathi Sripada, School of Vedic Sciences, MIT-ADT University
Sudeshna Das, IIT Kharagpur
Sukhada, IIT (BHU), Varanasi
Sunil Kumar Kopparapu, Tata Consultancy Services Ltd., India.
Surangika Ranathunga, University of Moratuwa
Surmila Thokchom, NIT Meghalaya
Svetla Koeva, IBL, Bulgarian Academy of Sciences
Swapnil Hingmire, TCS Research, India
Tadashi Nomoto, National Institute of Japanese Literature
Tanik Saikh, IIT Patna, India
Tanmoy Chakraborty, IIIT Delhi India
Tapas Nayak, TCS Research and Innovation
Thiyam Susma Devi, IIT Guwahati, India
Tollef Jørgensen, NTNU (DART, IDI)
Vikas Verma, KCL IMT Jalandhar
Wes Robbins, Montana State University
Ye Liu, Mercedes-Benz AG, Sindelfingen, Germany
Zhongkai Sun, Amazon Alexa AI

Organized by

National Institute of Technology Silchar   Natural Language Processing Association of India

Sponsored by

Samsung

Table of Contents

Conference Program

Day 1: Friday, December 17, 2021

10:00 AM–11:00 AM **Inaugural Ceremony**

11:30 AM–12:30 PM **Keynote Lecture 1:** Phrases and Self-Supervision in Neural Machine Translation
Prof. Josef van Genabith
Session Chair: Sivaji Bandyopadhyay

12:30 PM–02:30 PM **BREAK**

02:30PM–06:00PM **Technical Session I:** Machine Translation
Session Chair: Partha Pakray **Co-Chair:** Badal Soni

Constrained Decoding for Technical Term Retention in English-Hindi MT
Niyati Bafna, Martin Vastl and Ondřej Bojar

Named Entity-Factored Transformer for Proper Noun Translation
Kohichi Takai, Gen Hattori, Akio Yoneyama, Keiji Yasuda, Katsuhito Sudoh and Satoshi Nakamura

Multi-Task Learning for Improving Gender Accuracy in Neural Machine Translation
Carlos Escolano, Graciela Ojeda, Christine Basta and Marta R. Costa-jussa

Small Batch Sizes Improve Training of Low-Resource Neural MT
Àlex Atrio and Andrei Popescu-Belis

lakṣyārtha (Indicated Meaning) of Śabdavyāpāra (Function of a Word) framework from kāvyaśāstra (The Science of Literary Studies) in Samskṛtam : Its application to Literary Machine Translation and other NLP tasks
Sripathi Sripada, Anupama Ryali and Raghuram Sheshadri

EduMT: Developing Machine Translation System for Educational Content in Indian Languages
Ramakrishna Appicharla, Asif Ekbal and Pushpak Bhattacharyya

Assessing Post-editing Effort in the English-Hindi Direction
Arafat Ahsan, Vandan Mujadia and Dipti Misra Sharma

An Experiment on Speech-to-Text Translation Systems for Manipuri to English on Low Resource Setting
Loitongbam Sanayai Meetei, Laishram Rahul, Alok Singh, Salam Michael Singh, Thoudam Doren Singh and Sivaji Bandyopadhyay

Day 1: Friday, December 17, 2021 (continued)

Day 1: Friday, December 17, 2021 (continued)

Day 1: Friday, December 17, 2021 (continued)

Co-attention based Multimodal Factorized Bilinear Pooling for Internet Memes Analysis
Gitanjali Kumari, Amitava Das and Asif Ekbal

06:00 PM–06:30 PM **TEA BREAK**

06:30 PM–07:30 PM **Keynote Lecture 2:** Mental Health as an Application Area for Natural Language Processing: Prospects and Challenges
Prof. Philip Resnik
Session Chair: Dipti Misra Sharma

07:30 PM–07:45 PM **BUFFER**

07:45 PM–08:45 PM **Keynote Lecture 3:** Multimodal AI: Understanding Human Behaviors
Dr. Louis-Philippe Morency
Session Chair: Sivaji Bandyopadhyay

Day 2: Saturday, December 18, 2021

9:00 AM–12:00 PM **Technical Session IV:** Machine Learning in NLP
Session Chair: Sudeshna Sarkar **Co-Chair:** Anupam Biswas

How effective is incongruity? Implications for code-mixed sarcasm detection
Aditya Shah and Chandresh Maurya

Contrastive Learning of Sentence Representations
Hefei Qiu, Wei Ding and Ping Chen

Classifying Verses of the Quran using Doc2vec
Menwa Alshammeri, Eric Atwell and Mohammad Alsalka

ABB-BERT: A BERT model for disambiguating abbreviations and contractions
Prateek Kacker, Andi Cupallari, Aswin Subramanian and Nimit Jain

Training data reduction for multilingual Spoken Language Understanding systems
Anmol Bansal, Anjali Shenoy, Krishna Chaitanya Pappu, Kay Rottmann and Anurag Dwarakanath

Leveraging Expectation Maximization for Identifying Claims in Low Resource Indian Languages
Rudra Dhar and Dipankar Das

Day 2: Saturday, December 18, 2021 (continued)

Day 2: Saturday, December 18, 2021 (continued)

03:30 PM–03:45 PM **BREAK**

03:45 PM–06:00 PM **Technical Session VII:** Computational Psycholinguistics/Sentiment Analysis and Emotion Recognition
Session Chair: Dipankar Das **Co-Chair:** Ujwala Baruah

Sentiment Analysis For Bengali Using Transformer Based Models
Anirban Bhowmick and Abhik Jana

IndicFed: A Federated Approach for Sentiment Analysis in Indic Languages
Jash Mehta, Deep Gandhi, Naitik Rathod and Sudhir Bagul

An Efficient BERT Based Approach to Detect Aggression and Misogyny
Sandip Dutta, Utso Majumder and Sudip Naskar

How vulnerable are you? A Novel Computational Psycholinguistic Analysis for Phishing Influence Detection
Anik Chatterjee and Sagnik Basu

Aspect Based Sentiment Analysis Using Spectral Temporal Graph Neural Network
Abir Chakraborty

Using Random Perturbations to Mitigate Adversarial Attacks on Sentiment Analysis Models
Abigail Swenor and Jugal Kalita

Retrofitting of Pre-trained Emotion Words with VAD-dimensions and the Plutchik Emotions
Manasi Kulkarni and Pushpak Bhattacharyya

Pre-conference: Thursday, December 16, 2021

10:00 AM – 12:30 PM **Doctoral Consortium Session**

Session Chair: Radhika Mamidi **Co-Chair:** Samar Hussain

Bypassing Optimization Complexity through Transfer Learning & Deep Neural Nets for Speech Intelligibility Improvement
Ritujoy Biswas

Design and Development of Spoken Dialogue System in Indic Languages
Shrikant Malviya

Post-Conference: Sunday, December 19, 2021

10:00AM – 05:00 PM **Tools, Demo and Shared Tasks**
Chair: Partha Pakray **Co-Chair:** Vishal Goyal

FinRead: A Transfer Learning Based Tool to Assess Readability of Definitions of Financial Terms
Sohom Ghosh, Shovon Sengupta, Sudip Naskar and Sunny Kumar Singh

Demo of the Linguistic Field Data Management and Analysis System - LiFE
Siddharth Singh, Ritesh Kumar, Shyam Ratan and Sonal Sinha

Text Based Smart Answering System in Agriculture using RNN
Raji Sukumar, Hemalatha N, Sarin S and Rose Mary C A

Image2tweet: Datasets in Hindi and English for Generating Tweets from Images
Rishabh Jha, Varshith Kaki, Varuna Krishna Kolla, Shubham Bhagat, Parth Patwa, Amitava Das and Santanu Pal

Constrained Decoding for Technical Term Retention in English-Hindi MT

Niyati Bafna*, **Martin Vastl***, **Ondřej Bojar**
Charles University, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics
niba00005@stud.uni-saarland.de, martin.vastl2@gmail.com, bojar@ufal.mff.cuni.cz

Abstract

Technical terms may require special handling when the target audience is bilingual, depending on the cultural and educational norms of the society in question. In particular, certain translation scenarios may require "term retention" i.e. preserving of the source language technical terms in the target language output to produce a fluent and comprehensible code-switched sentence. We show that a standard transformer-based machine translation model can be adapted easily to perform this task with little or no damage to the general quality of its output. We present an English-to-Hindi model that is trained to obey a "retain" signal, i.e. it can perform the required code-switching on a list of terms, possibly unseen, provided at runtime. We perform automatic evaluation using BLEU as well as F1 metrics on the list of retained terms; we also collect manual judgments on the quality of the output sentences.

1 Introduction and Motivation

It is common for bilingual or multilingual speakers to borrow technical terms from other, usually high resource, languages into their native language. This may be for several reasons, e.g. the technical term in the high resource language may be much more popular and therefore better understood, or the required term may simply not exist in the language in question. This is very common, for example, in Indian languages, where the language of education is frequently different from the regional native language.

We can imagine, therefore, a scenario which requires the automatic translation of text or speech, with the constraint that a given list of English domain words appear untranslated in the Hindi output. Essentially, this can be seen as a special case of constrained decoding with a given source-target terminology. We make the assumption that the user knows the terms to be retained at run time, and can provide this information to the system before translating the sentence.[2]

*Equal contribution by these authors.

2 Previous Work

The idea of constrained decoding has been recognized as useful in several works (Hokamp and Liu, 2017; Chatterjee et al., 2017; Hasler et al., 2018; Dinu et al., 2019; Jon et al., 2021). Usually, the constraints are in the form of a terminology list, as in the above works. To our knowledge, this is the first study on combining this concept with introducing code-switching[3] (CS) into the output for a multilingual educational or technical setting.

3 Approach

We set up an end-to-end supervised learning scenario aimed at teaching the model to perform term retention. The basic idea is to train a machine translation model to obey a "signal", that we can then provide at run time on selected words. It is easy to see that such a model (the "tagged" model) would be independent of domain and could in theory perform term retention on any term for which the signal was provided. We also train a simple baseline for comparison; the baseline model sees the same training data as the tagged model, but does not receive any signal that would be highlighting the terms to retain. Therefore, given input at run time, it must rely on past exposure on the specific terms and their (non-)translation to perform term retention.

We provide the mentioned signal in the form of tags i.e. *<REW>* and *</REW>* tags (standing

[2]We do not, however, assume that we have this information while training, since it would be expensive and unviable to retrain such a model every time for a new setting and/or new domain vocabulary. In this study, we work with English-Hindi MT.

[3]Linguists sometimes make a difference between the terms code-switching and code-mixing; in this paper, they are used interchangeably.

Source sentence: *You need to install these Python libraries.*
Term list: Python, libraries
Input to the system: *You need to install these <REW> Python </REW> <REW> libraries </REW>*
Desired output: **आपको इन Python libraries को स्थापित करना चाहिए**

Figure 1: Example input to and desired output from the system

Dataset	Total sentences	Sentences with CS
Train	250 700	123 274
Development	10 247	5 064
Test seen	5 000	5 000
Test unseen	768	768
Test w/o CS	500	0

Table 1: Types of datasets. CS Sentences: sentences with introduced code-switching. "Test seen": sentences with terminology that were all seen during the training, "Test unseen": sentences with terminology that were never seen during the training as retained words. Test w/o CS: sentences with no terminology constraints.

for "retained English word") to indicate that the enclosed term shall be retained during the translation, see Figure 1. This approach can be used in any type of transformer-based translation system and therefore can be implemented with little to no effort in current systems.

4 Synthetic Data Creation

We used HindEnCorp 0.5 (Bojar et al., 2014) data set and we split it into multiple parts as seen in Table 1. We adapt pre-existing English-Hindi parallel data so that it manifests term retention on the target while remaining coherent and grammatical. We leverage the fact that our parallel corpus already contains many instances of simple transliteration equivalents, such as names of people, places, organizations, etc. We thus interpret the target sentence as "retaining" the transliterated word, while being perfecty grammatical.[4]

4.1 Identifying Transliterations

Given the parallel corpus, we need to identify pairs of transliterated words in each English-Hindi sentence pair. We first find the word level alignments[5] in source-target pairs, using GIZA++ (Och and Ney, 2003). Then for each aligned word pair, we check for transliteration using a normalized edit distance threshold.[6] We define our normalized edit distance as:

$$NED(s,t) = \frac{edit\_distance(s,t)}{max(length(s), length(t))}$$

calculated between the English word and the Hindi word transliterated into Latin script.[7] Eyeballing the resulting pairs, we see that the alignment step along with this threshold results in near perfect accuracy. This method gives us a total of 269095 transliteration pairs in the whole corpus.

Once a transliteration pair is identified in the training corpus, we simply replace the target side Devanagari word with the Latin-script source word, resulting in an instance of term retention. The original sentence pair is no longer used in the training of the tagged model.

5 Model

We used a transformer-based model (Vaswani et al., 2017) with vocabulary size of 32000 tokens and with hyperparameters as described in The University of Edinburgh's Neural MT Systems for WMT17 (Sennrich et al., 2017) for both of our models. We used MarianMT framework (Junczys-Dowmunt et al., 2018) to train the models; we let the model train until the BLEU score (Papineni et al., 2002) did not improve on the development set for 5 epochs. We then selected the model with the highest BLEU score as the model used for later experiments. The change of BLEU score on the

[4]Although more sophisticated approaches to synthetic code-switched data creation may be better suited for other tasks, we find that this approach is sufficient for our needs. This may be because term retention is in fact required to be performed on similar words i.e. named entities or domain terms that behave similarly to named entities.

[5]The idea is that the target transliterated word must "come from" or be aligned with the source word, assuming a correct word alignment.

[6]We use a Python transliteration tool `https://pypi.org/project/indic-transliteration/`

[7]The threshold was tuned over a small subset of the Xlit-Crowd: Hindi-English Transliteration Corpus (Khapra et al., 2014): using this corpus, we found the edit distance between the English source words and the "true" transliterations which were back-transliterated into Latin script. For the final experiment, we used the threshold of 0.5.

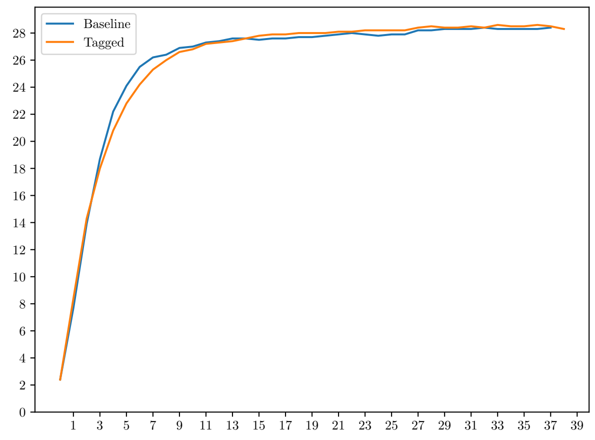


Figure 2: BLEU score on development set per epoch

Model	Seen	Unseen	Without CS
Baseline	**28.7**	16.8	**22.4**
Tagged	27.2	**17.5**	21.9

Table 2: BLEU score on test set

development set per epoch is in Figure 2. It can be seen that the BLEU scores for both of the models are comparable and they train for a similar number of epochs.

6 Automatic Evaluation

There are two components of model performance:

- Retention of marked terminology
- Overall coherence and fluency

For the former, we calculate precision, recall, and F1 over the gold retained set of words and the set of retained terms in the output. Our evaluation script compares the system output with the list of terms that should be untranslated in the given sentence. Precision is the ratio of term occurrences in the system output that were anticipated in the reference, out of all produced Latin terms. Recall is the ratio of term occurrences produced by the system out of all term occurrences anticipated by the reference. For the latter, we use BLEU score.

The BLEU scores on test sets can be seen in Table 2. The baseline model is slightly better on the seen test set, while the tagged approach outperforms the baseline model on the unseen test set. On the "Without CS" test, the baseline model still (incorrectly) produces English; however, while the tagged model does not do this, it often produces different and sometimes incorrect Hindi phrasing for these words as compared to the reference, resulting in an overall lower BLEU score. A possible explanation for this observation is that the tagged model has to learn to use the given signal at proper places which can damage its performance. On the other hand on the unseen dataset, the tagged model receives explicit information to retain the term and therefore outperforms the baseline model. Results for the retention metric can be seen in Table 3 and Table 4.

Model	Precision	Recall	Micro F1
Baseline	0.43	0.63	0.51
Tagged	**0.88**	**0.88**	**0.88**

Table 3: Retention results on seen test set

Model	Precision	Recall	Micro F1
Baseline	0.08	0.25	0.13
Tagged	**0.51**	**0.85**	**0.64**

Table 4: Retention results on unseen test set

It can be seen that the tagged approach outperforms the baseline model on both the unseen and seen test set, demonstrating that it indeed learns to obey the provided signal, instead of simply relying on previous exposure as the baseline does.[8]

7 Manual Evaluation

We also performed a manual evaluation to complement the BLEU score. This evaluation was solely for the purpose of judging the quality of the final output regardless of whether the model managed to retain the required words or not.

7.1 Design

We provide the annotators with the *spoken* form of the candidate translation, rather than asking them to read the script-mixed output. There are two reasons for this: (1) we do not want the annotators to be affected by seeing or not seeing Latin script, (2) the spoken form is the more natural setting in which code mixing occurs.

[8]Note that the drop in performance of the tagged model in the unseen test F1 score indicates that it is not wholly independent as yet of the terminology it has been exposed to.

Further, in order to ensure blind evaluation of the Baseline vs. Tagged system, we needed to control for the fact that the Tagged system has a higher tendency to retain words in the Latin script. Since the user may be unfairly biased one way or the other when judging between sentences with different numbers of code-switched words, we decided to select the test sentences in a controlled manner, depending on the number and nature of Latin-spelled (i.e. English) words in the output.

The test set partitions are listed as columns in Tables 5 and 6: "Same # of En words" is the group of test sentences where the Baseline and Tagged translated outputs have the same number of English terms, thus controlling for bias for or against a translation simply because it has more English. In total, there were 5 such sentences each scored 3 times, so we collected 15 judgements on this partition. For instance in Table 5, we see that the tagged model was selected as better by 7 judgements and in 4 cases, it tied with the baseline. "Same set of En words" takes this a step further: it is the group of sentences where both model outputs have exactly the same English words in them; of course, they may (and do) differ in the rest of the sentence structure, Hindi wording, etc. Note that selecting sentences with a comparable number of terms English in them as we do results in an inherent advantage for the baseline model: since the baseline model can code-switch when it chooses rather than according to an external signal, it is more likely to choose convenient situations with globally better translations. This is the reason for the "Random" test set (the last column in Tables 5 and 6); i.e. sentences picked randomly, regardless the output of each system, which are intended to judge the average quality of the baseline and tagged against each other, even though these judgments are vulnerable to the biases discussed above.

In the manual evaluation, we gave 3 native Hindi speakers, also fluent in English, the source text and recordings of the translations. The goal of the annotation was a three-way judgment: whether the first translation was better, the second was better, or both were equivalent in quality.

7.2 Results and Analysis

Our manual test set covers a total of only 26 sentences, split equally between outputs from the seen and unseen test sets;[9] it is intended more for giving a qualitative sense of the comparison. Broadly, the evaluators considered the tagged outputs roughly comparable to the baseline in terms of coherence and quality, see Tables 5 and 6. Across both test sets, the Baseline model outputs were considered better 33% of the time (26 of 78 judgments), the Tagged model outputs were considered better 32% of the time (25 judgments), and the outputs were considered roughly equivalent in quality in the remaining 35% of the judgments.

We investigated the following questions:

- Do the models perform better on seen words than on unseen words?

	Same # of En words	Same set of En words	Random	$\sum$
Baseline	4	5	3	12
Tagged	7	4	1	12
Equal	4	6	5	15
$\sum$	15	15	9	39

Table 5: Manual test judgments for seen test set. Overall, the set contains 13 sentences from the seen test set, leading to the total of 39 judgments over 3 annotators. For example, we had 3 sentences (and therefore 9 total judgments) in the randomly selected group of sentences ("Random"); of these 9 judgments, 4 preferred the baseline model, 1 the tagged model, and 5 judgments saw the baseline and tagged outputs as equal in terms of overall quality.

	Same # of En words	Same set of En words	Random	$\sum$
Baseline	3	8	3	14
Tagged	7	4	2	13
Equal	5	3	4	12
$\sum$	15	15	9	39

Table 6: Manual test judgments for unseen test set. This test set again contains 13 sentences from the unseen test set, so a total of 39 judgments over 3 annotators is collected. The columns have the same meaning as in Table 5.

[9]This is because of the demanding procedure involving sentence recordings.

In the manual evaluation, we observed that the models dip in fluency around the segments with introduced English words. For example, there is a lack of syntactic agreement, or the model loses the thread of the sentence.

Tagged: *आवश्यक packages हटाया जाएगा।
(*Essential packages will be$_{sg}$ removed$_{sg}$)

In this example, we need the plural inflection of the verb phrase "हटाया जाएगा।" (will be removed). We see these instances both in the seen and unseen test sets; however, on the whole, the models are able to keep track of the source sentences a little better with the seen test set.

- Why does tagged do better than baseline in sentences where the same number of English words was produced in the output?

  The baseline model is worse at retaining fluency around code-switched words, especially in the unseen test set. While the tagged model also shows this tendency, it manages to translate the shorter instances correctly. With longer sentences, it is performing equally bad, especially in the unseen test set.

The "random" test set is intended to take a look at the average outputs of the models, not controlled for the number of English words in them. Here, the models perform similarly, but users differ in their preferences regarding the presence of English words.[10] Overall, the qualitative assessment yields that the tagged model performs on par with the baseline with respect to fluency, and of course much better at the retention task.

8 Conclusion

The task of applying terminology constraints while dealing with code-switched text seems especially important in current multilingual educational and other settings. We present a simple technique that can adapt a vanilla transformer-based MT tool for performing this task, by synthesizing training data that exhibits term retention. We demonstrate that our model performs well on unseen terminology, and that its general translation quality is not damaged. Future research should consider using code-switched parallel corpora, either for training or fine-tuning, in order to teach the models the various nuances of natural human code-mixing.

[10]For example, in a sentence that only differs in the fact that a word is in English in the first sentence and in the Hindi form in the second sentence, annotators apply their preferences.

Acknowledgements

This project has received funding from the grants H2020-ICT-2018-2-825303 (Bergamot) of the European Union and 19-26934X (NEUREM3) of the Czech Science Foundation. Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA CZ LM2018140) supported by the Ministry of Education, Youth and Sports of the Czech Republic and datasets come from the Lindat Repository supported by the Ministry of Education, Youth and Sports of the Czech Republic, Project No. LM2018101 LINDAT/CLARIAH-CZ.

References

Ondřej Bojar, Vojtěch Diatka, Pavel Straňák, Aleš Tamchyna, and Daniel Zeman. 2014. HindEnCorp 0.5. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Rajen Chatterjee, Matteo Negri, Marco Turchi, Marcello Federico, Lucia Specia, and Frédéric Blain. 2017. Guiding neural machine translation decoding with external knowledge. Association for Computational Linguistics.

Georgiana Dinu, Prashant Mathur, Marcello Federico, and Yaser Al-Onaizan. 2019. Training neural machine translation to apply terminology constraints. *arXiv preprint arXiv:1906.01105*.

Eva Hasler, Adrià De Gispert, Gonzalo Iglesias, and Bill Byrne. 2018. Neural machine translation decoding with terminology constraints. *arXiv preprint arXiv:1805.03750*.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. *CoRR*, abs/1704.07138.

Josef Jon, João Paulo Aires, Dusan Varis, and Ondrej Bojar. 2021. End-to-end lexically constrained machine translation for morphologically rich languages. *CoRR*, abs/2106.12398.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Mitesh M Khapra, Ananthakrishnan Ramanathan, Anoop Kunchukuttan, Karthik Visweswariah, and Pushpak Bhattacharyya. 2014. When transliteration met crowdsourcing: An empirical study of transliteration via crowdsourcing using efficient, non-redundant and fair quality control. In *LREC*, pages 196–202. Citeseer.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

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.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. 2017. The University of Edinburgh's neural MT systems for WMT17. In *Proceedings of the Second Conference on Machine Translation*, pages 389–399, Copenhagen, Denmark. 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. *CoRR*, abs/1706.03762.

Named Entity-Factored Transformer for Proper Noun Translation

Kohichi Takai†‡ Gen Hattori‡ Akio Yoneyama‡
Keiji Yasuda†‡♦ Katsuhito Sudoh† Satoshi Nakamura†

†NARA Institute of Science and Technology ‡KDDI Research, Inc ♦MINDWORD, Inc
{ ko-takai , ge-hattori , yoneyama }@kddi-research.jp
{ ke-yasuda , sudoh , s-nakamura }@dsc.naist.jp

Abstract

Subword-based neural machine translation is almost free from out-of-vocabulary (OOV) words. However, it does not always work well for composing proper nouns. We propose a method to use Named Entity (NE) features with Factored Transformer for accurate proper noun translation. The NE features are extracted from NE recognition on input sentences. Our experimental results showed the proposed method outperformed the baseline subword-based Transformer in BLEU and proper noun translation accuracy.

1 Introduction

Recent advances in neural machine translation (NMT) have made machine translation (MT) systems useful in practical applications. However, translation of proper nouns still remains difficult in spite of its importance in practice. Proper nouns are sometimes processed as out-of-vocabulary (OOV) words in MT systems due to the limitation of the vocabulary size and data sparseness. Approaches for proper noun translation can be divided roughly into two approaches: the use of hand-crafted bilingual lexicon as the external knowledge and the use of subwords.

The former approach uses a bilingual proper noun dictionary to translate proper nouns. Okuma et al (2008) proposed replacement-based proper noun translation. Their method uses a bilingual dictionary whose entries are associated with proper noun classes to replace a proper noun with another surrogate proper noun that frequently appears in the training corpus. Another method called lexically constrained decoder (LCD) (Hokamp et al., 2017) guarantees that proper nouns are translated into the target language sentence constrained by a bilingual dictionary (Chen et al., 2020, Chousa et al., 2021). It extends the beam search algorithm to find the hypothesis that contains all of the proper nouns (Hokamp et al., 2017). The dictionary-based approach works well only if the proper nouns to be translated are included in the bilingual dictionary and requires efforts for developing the dictionary.

In NMT, the subword-based approach is widely used. Sennrich et al (2016) proposed the use of subwords to decompose a word into shorter units. The method decreases the number of OOV words and keeps the translation quality if input sentences include OOV words. However, the subword-based NMT does not always work on a proper noun translation due to wrong compositions of subword translations.

In this paper, we propose a method for NMT focusing on the proper noun translation using Factored Transformer with named entity (NE) features. The proposed method only uses a parallel corpus and an NE recognition (NER) model as external knowledge.

2 Related Work

2.1 Factored NMT

Factored NMT (García-Martínez et al., 2016) integrates linguistic information into an NMT decoder. It decomposes morphological and grammatical features of a word into factors. Jordi et al. (2019) proposed Factored Transformer as an extension of Transformer (Vaswani et al., 2017) for low-resource NMT. The outputs from its subword and factor embedding layers are combined.

2.2 Named Entity Recognition

NER identifies and classifies proper nouns in a sentence. Recent studies in NER use neural networks as well. Huang et al., (2015) used Long Short-Term Memory (LSTM) and Conditional Random Field (CRF). Arkhipov et al., (2019) used BERT (Devlin et al., 2018) for NER. BERT utilizes a multilayer bidirectional transformer encoder which can learn deep bi-directional representations and can be fine-tuned for various NLP tasks later.

With respect to named entities, the use of a class-based language model was proposed to solve the problem of data sparseness in the field of automatic speech recognition (ASR) research field (Yamamoto et al., 1999, 2004). This idea was extended to MT (Tonoik et al., 2005, Yasuda et al., 2017). This approach improved the translation performance for unknown and low-frequency words by using high-frequency surrogate words in the same category.

The main focus of this paper is proper noun translation in the subword-based NMT using Factored Transformer and NE features from NER without a bilingual dictionary.

3 Proposed Method

We propose the use of NE features as linguistic factors of Factored Transformer for accurate proper noun translation.

3.1 Named Entity Feature Vector

NE features obtained from NER on a source language sentence are injected into the embedding or encoder layer, as a factor. We can use two types of NE features: a one-hot NE vector and an NE probability distribution vector. We extract the NE features from an input source language sentence by the following steps.

1. Apply word segmentation into an input source language sentence.
2. Apply subword segmentation onto the word-segmented input.
3. Apply part-of-speech (POS) tags to the word-segmented input using a POS tagger.
4. Recognize NE in the POS-tagged input sentence to obtain one-hot NE vectors or NE probability distribution vectors.
5. Align those NE feature vectors with the subwords composing corresponding words.

Here, a one-hot NE vector represents a 1-best NE category while an NE probability distribution vector represents the ambiguity of NE categories.

3.2 Factored Transformer Architecture

We propose a Factored Transformer model that uses two factors: subwords and NE features. We present two types of NE features and the two model variants in factor-injecting layers, as shown in Fig.1.

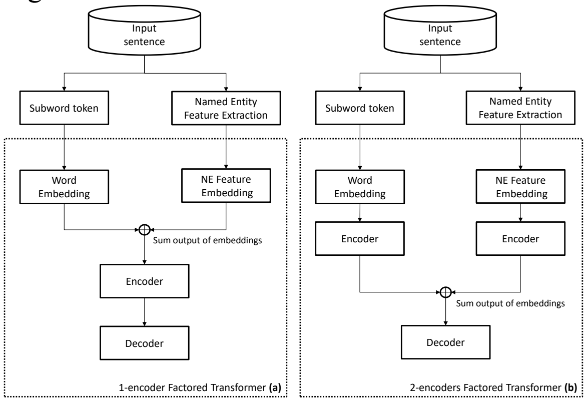


Figure 1: 1-encoder (a) and 2-encoders models (b)

1-encoder model (Fig. 1(a)):
Each factor has its own embedding layer. The embedding vectors are summed up together with the corresponding positional encoding vector and sent to the following encoder layer. The rest of the model remains unchanged from the *vanilla* Transformer.

2-encoders model (Fig. 1 (b)):
Each factor has its own encoder in addition to the embedding layer. The outputs from the encoders are summed up and used as encoder outputs. The rest of the model remains unchanged.

4 Experimental Settings

We conducted Japanese-to-English and English-to-Japanese MT experiments to compare the performance of the proposed method with a standard NMT method.

4.1 Named Entity Recognition Model

For the Japanese-to-English experiments, we used an NER model based on a pre-trained BERT model and fine-tuned it using Japanese NER training data generated by using the method presented by Takai et al., (2018). Table 1 shows the detailed parameter settings of the NER model.

parameter	**mini batch size**	**epoch**	**optimizer**
BERT-NER	**32**	**4**	**Adam**

Table 1: Detail of NER Hyperparameter

For the English-to-Japanese experiments, we used the NER module of Stanza[1]. In the experiments, the Japanese NER model had 33 categories, and the English NER model had 77 categories.

As a bilingual corpus of NER training data on automatic construction method (Takai et al., 2018) in Japanese-to-English experiments, we used 10 million Japanese and English part sentences of JParaCrawl. We extracted sentence pairs including proper nouns by using tagger and POS. Here, we used Sudachi[2] for Japanese morphological analysis to find proper nouns. We chose 1,000 sentence pairs containing proper nouns for the NER training, based on the sentence pair scores (Morishita et al., 2020).

4.2 MT Models

We used Transformer and Factored Transformer models for NMT, with 6-layer encoders and decoders. The configurations of the models and their training were mostly the same as those of the *vanilla Transformer*, but we used different settings on the hyperparameters as shown in the following Table 2.

directions	max token size	max epoch
J-E	7,300	60
E-J	7,300	33

Table 2: Details of NMT hyperparameters

We used SentencePiece (Kudo et al., 2018) with a subword unigram model for the subword tokenization. We used Sudachi and Moses[3] as Japanese and English POS taggers.

4.3 Training and Dev. Data for MT Models

Details of the corpus for the NMT models are shown in Table 3. For the Japanese-to-English experiments, we used a part of 10 million Japanese-to-English sentence pairs in JParaCrawl (Morishita et al., 2020) for the training of the NMT models. We chose 160,000 sentence pairs that contain proper nouns, have sentence pair scores higher than 0.786, and shorter than 250 subwords. For the English-to-Japanese experiments, we used all the 10 million sentence pairs in JParaCrawl as a training data set due to the effectiveness of the different conditions from the Japanese-to-English one: language pairs and amount of training data. WMT 2020 development set[4] was used as the development set for all the NMT models.

	direc tion	# of sentences	# of subwords	# of uniq subwords
Train	J-E	159,888	5,318,140	10,073
Dev		10,000	333,933	9,941
Train	E-J	10,116,570	332,520,88 8	47,087
Dev		1,998	65,649	6,873

Table 3: Details of corpus size

4.4 Evaluation Data

Details of the evaluation data are shown in Table 4.

For the Japanese-to-English, we used an evaluation dataset of 271 sentences containing a single proper noun. It was collected through field experiments with taxis in Japan and was translated manually. The data consisted of conversations between taxi drivers and travelers. For the English-to-Japanese task, we used WMT 2020 Test set.

direction	# of sentences	# of subwords	# of uniq subwords
J-E	271	4,258	646
E-J	1,000	32,696	5,171

Table 4: Details of evaluation data size

4.5 Compared Methods

We compared the following NMT models:

- Transformer (baseline)
- Proposed methods with the combination of the model architecture and the NE feature vector representations:
    - 1-encoder / 2-encoders
    - NE one-hot vector / NE probability distribution vector

4.6 Evaluation metrics

We used BLEU (Papineni et al., 2002) as a translation quality metric. We also evaluated proper noun translation accuracy (PRPacc); i.e., the percentage of proper noun words that correctly translated over the entire test set.

[1] http://nlp.stanford.edu/software/stanza/1.2.2/en/ner/ontonotes.pt

[2] https://github.com/WorksApplications/Sudachi

[3] http://www.statmt.org/moses/

[4] http://www.statmt.org/wmt20/translation-task.html

5 Results

Table 5 shows the results. In Japanese-to-English, the proposed 1-encoder models were worse than the baseline, but the 2-encoders models outperformed the baseline. The results by the 2-encoder model with NE probability distributions showed the best performance, outperformed the baseline by 9.6 points in PRPacc and 2.5 points in BLEU. In English-to-Japanese, however, the 1-encoder models outperformed the baseline. The improvement in BLEU and PRPacc was smaller than that in Japanese-to-English. This may be due to the difference in the training data sizes; the English-to-Japanese MT models were trained using the 60 times larger parallel corpus. Another possible reason is the difference in the degrees of difficulty in these domains; WMT News task would be more difficult than the taxi conversation.

With respect to proper noun translation, the lack of a specific treatment of proper noun translation in the baseline resulted in worse performance than the proposed method. Translation examples are shown in Table 5. The 2-encoders models worked well on two types of proper nouns: the non-compositional proper noun of Table 6 (1), and the combination with proper nouns and general noun of Table 6 (2). This result can be assumed that the factor of NE feature vector directly works on the proper noun translation better in the near decoder.

As shown in Table 6, 2-encoders with NE probability distributions have better performance than 2-encoders with NE one-hot vector performance. Expression of the ambiguity of proper nouns in the NE probability distributions method influent on not only proper noun translation but also the surrounding words of a proper noun.

NMT Model	**NE Feature**	**PRPacc(%)**		**BLEU**	
		J-E	**E-J**	**J-A**	**E-J**
vanilla (baseline)	-	56.1	46.5	11.4	17.5
1-encoder	One-hot	43.2	**50.1**	10.1	**18.8**
2-encoders		63.5	47.5	13.8	17.8
1-encoder	Probability distributions	53.5	49.5	10.9	18.4
2-encoders		**65.7**	46.7	**13.8**	17.6

Table 5:Proper noun accuracy and BLEU in J-E task / E-J task

6 Conclusions

We proposed a method to enhance accurate proper noun translation using subword-based NMT by Factored-Transformer and NE features. The NE feature vectors are injected into Factored Transformer model as factors together with subwords. In the Japanese-to-English experiments using a small bilingual training corpus, the proposed method using the best NE feature vector outperformed the baseline sub-word-based transformer model by more than 9.6 points in proper noun accuracy and 2.5 points in the BLEU score. It also showed some improvements in the English-to-Japanese experiments using a large-scale bilingual corpus.

In future work, we will work on automatic clustering of proper nouns instead of given NE categories.

(1) Input sentence: 山の上に**岐阜城**があります (**Gifu Castle** is on top of the mountain.)		
vanilla	-	there are castle on the mountains above the mountains.
1-encoder	One-hot	mount Huangshan is a mountains above the altitude.
2-encoders		there are **Gifu Castle** on the mountains of the mountains.
1-encoder	Probability distributions	In the mountains, **Gifu Castle** is located above the top.
2-encoders		there is **Gifu Castle** on the top of the mountain.
(2) Input sentence: この城は**豊臣秀吉**が作りました (This castle was built by **Toyotomi Hideyoshi**.)		
vanilla	-	this castle was created by an excellent Japanese castle.
1-encoder	One-hot	this castle was created by yoshino hideyoshi hideyoshinori.
2-encoders		this castle of this castle was created by **toyotomi hideyoshi**.
1-encoder	Probability distributions	this castle was created by minister **toyotomi hideyoshi.**
2-encoders		this castle was created by **toyotomi hideyoshi**.

References

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. 2017. *Attention is all you need.* In Advances in neural information processing systems, pages 5998–6008.

Chris Hokamp and Qun Liu. 2017. *Lexically constrained decoding for sequence generation using grid beam search.* In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1535–1546.

Guanhua Chen, Yun Chen, Yong Wang and Victor O.K. Li. 2020. *Lexical-constraint-aware neural machine translation via data augmentation.* In Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, pages 3587–3593. International Joint Conferences on Artificial Intelligence Organization. Main track.

Hideo Okuma, Hirofumi Yamamoto and Eiichiro Sumita. 2008. *Introducing a translation dictionary into phrase-based smt.* The IEICE Transactions on Information and Systems 91-D pages 2051–2057.

Hirofumi Yamamoto, Hiroaki Kokubo, Genichiro Kikui, Yoshihiko Ogawa and Yoshinori Sagisaka. 2004. *Out-of-vocabulary word recognition with a hierarchical language model using multiple markov model.* In IEICE D-II 87(2), pages. 2104–2111.

Hirofumi Yamamoto and Yoshinori Sagisaka. 1999. *Multi-Class Composite N-gram based on connection.* In ICASSP, pages. 533-536

Huang Zhiheng, Xu Wei and Yu Kai. 2015. *Bidirectional LSTM-CRF models for sequence tagging.* arXiv preprint arXiv:1508.01991.

Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. 2018. *bidirectional transformers for language understanding.* arXiv:2004.08053v2

Jordi Armengol-Estapé, Marta R. Costa-jussà and Carlos Escolano 2020. *Enriching the Transformer with Linguistic Factors for Low-Resource Machine Translation.* arXiv preprint arXiv: arXiv:2004.08053v2.

Katsuki Chousa and Makoto Morishita. 2021. *Augmentation Improves Constrained Beam Search for Neural Machine Translation* The 8th Workshop on Asian Translation (WAT 2021)

Keiji Yasuda, Panikos Heracleous, Akio Ishikawa, Masayuki Hashimoto, Kazunori Matsumoto and Fumiaki Sugaya. 2017. *Building a location dependent dictionary for speech translation systems.* In CICLING.

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 (ACL).

Kohichi Takai, Gen Hattori, Keiji Yasuda, Panikos Heracleous, Akio Ishikawa, Kazunori Matsumoto and Fumiaki Sugaya. 2018. *Automatic Method to Build a Dictionary for Class-based Translation Systems* In CICLING.

Makoto Morishita, Jun Suzuki and Masaaki Nagata. 2020. *JParaCrawl: A Large Scale Web-Based English-Japanese Parallel Corpus.* In Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages. 3603-3609

Masatsugu Tonoike, Mitsuhiro Kida, Toshihiro Takagi, Yasuhiro Sasaki, Takehito Utsuro and Satoshi Sato. 2005. *Translation estimation for technical terms using corpus collected from the web.* In: Proceedings of the Pacific Association for Computational Linguistics, pages. 325–331.

Mercedes García-Martínez, Loïc Barrault and Fethi Bougares. 2016. *Factored neural machine translation.* CoRR, abs/1609.04621.

Mikhail Arkhipov, Maria Trofimova, Yuri Kuratov and Alexey Sorokin. 2019. *Tuning Multilingual Transformers for Named Entity Recognition on Slavic Languages* Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL) pages 89-93

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 (ACL), pages.1715-1725

Taku Kudo and John Richardson. 2018. *SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing.* In EMNLP demo, pages 66-71

.

Multi-Task Learning for Improving Gender Accuracy in Neural Machine Translation

Carlos Escolano[1] and **Graciela O. Jiménez**[1] and **Christine Basta**[1 2] and **Marta R. Costa-jussà**[1]
[1] TALP Research Center, Universitat Politècnica de Catalunya, Barcelona
[2] Institute of Graduate Studies and Research, Alexandria University, Egypt
{carlos.escolano,christine.raouf.saad.basta,marta.ruizs}@upc.edu
{graciela.ojeda}@estudiantat.upc.edu

Abstract

Machine Translation is highly impacted by social biases present in data sets, indicating that it reflects and amplifies stereotypes. In this work, we study mitigating gender bias by jointly learning the translation, the part-of-speech, and the gender of the target language with different morphological complexity. This approach has shown improvements up to 6.8 points in gender accuracy without significantly impacting the translation quality.

1 Introduction

In recent years, the awareness about the bias present in Machine Translation (MT) systems has increased in the scientific community, especially gender bias. Gender is manifested differently in languages; gender bias problems occur when translating between languages with +various levels of morphology. There is bias when the system tends to translate according to gender roles, even when there is no ambiguity (Prates et al., 2020).

Surveys (Sun et al., 2019; Blodgett et al., 2020; Costa-jussà, 2019; Savoldi et al., 2021) have recently shown the great efforts carried out by scientists towards resolving the problem of gender bias in NMT. Tagging and additional context approaches (Vanmassenhove et al., 2018; Moryossef et al., 2019; Basta et al., 2020) have shown an improvement in translation accuracy when translating from English to languages with more complex morphology. Domain adaptation techniques have proved to impact the performance of translation in (Saunders and Byrne, 2020). Debiased pre-trained word embeddings have been leveraged in (Escudé Font and Costa-jussà, 2019) and have shown improvement in Spanish translations. Gender bias is mainly attributed to the already present bias in the data used to train MT systems (Savoldi et al., 2021; Costa-jussà, 2019). Furthermore, it has been shown that models trained on these data tend to amplify further this bias (Zhao et al., 2018). In this sense, research done aims to avoid or reduce this amplification, such as fine-tuning techniques with gender-balanced dataset corpus (Costa-jussà and de Jorge, 2020) or annotating the source language words of the training data with the gender of the target language words (Stafanovičs et al., 2020). These techniques have shown promising results regarding gender accuracy.

In line with reducing the amplification of data, we propose to obtain a system capable of predicting the part-of-speech (pos) and the gender of the words of the target language, besides the translation. We expect that, by having more information about the output words, the system maintains the quality of the translation and is able to better predict the gender based on the context without falling so much into the stereotype. In general, the proposed configurations outperformed the baseline NMT system on gender prediction accuracy by up to 6.8% while retaining average translation performance.

2 Bias statement

Nowadays, we live in a more globalized and connected world, which leads society to use MT tools to communicate with different nationalities. The fact that standard translators present a gender bias harms society, helping to perpetuate certain stereotypes and prejudices. An example is the tendency of specific systems to generalize the different professions carried out by men and women (Prates et al., 2020). It is significantly more visible when one tries to translate from a gender-neutral language, such as English, to another with grammatical gender, such as Spanish. In the first type, nouns have no grammatical gender, while in the other type, there is gender inflection in nouns, adjectives, verbs, etc. Therefore, when translating from

English to Spanish, the system uses masculine or feminine inflections following stereotypes. Such stereotypical errors occur both when the context indicates gender explicitly and when it does not.

3 Background

In this section, we review the basics of multilinguality and multi-task learning in NMT, as well, as linguistic features and the framework to evaluate gender bias in MT. All these methodologies are later used in our proposed research.

Multilingual NMT. Transformers (Vaswani et al., 2017) have advanced NMT, giving the ability to pay more attention to multilingual NMT. The primary approach behind multilingual is to have the same model architecture to translate different language pairs (Firat et al., 2016; Johnson et al., 2017). Previous studies explore different design approaches for the model architecture, either partial sharing with shared encoder (Sen et al., 2019), shared attention (Firat et al., 2016), task-specific attention (Blackwood et al., 2018), shared parameters (Zhu et al., 2020), full model sharing (Johnson et al., 2017) or independent encoder-decoders without sharing (Escolano et al., 2021; Lu et al., 2018). In this paper, we adopt this architecture without sharing.

Multi-task learning NMT. Multi-task learning (Caruana, 1997) trains the model on several co-related tasks. This training can lead to generalized improved performance and facilitate sharing representations (Ruder, 2017). In the NMT context, injecting linguistic knowledge has been successful when training NMT with related tasks (POS tagging, dependency parsing). This linguistic injection can lead to improving NMT generalization and translation quality, especially in low-resource scenarios (Kiperwasser and Ballesteros, 2018; Zaremoodi and Haffari, 2018; Eriguchi et al., 2017). Our work depends on adopting linguistic knowledge through training multi-tasks, besides NMT. We trained our NMT model with POS prediction and gender tagging tasks.

Linguistic Features. Different linguistic features can be utilized for words' classification, such as part-of-speech (POS) in morphology. POS refers to the lexical category of words, defining different linguistic categories depending on the shared morphological categories between these words. Universal Dependencies (UD)[1] is a framework for consistent grammar annotation across different human languages (Nivre et al., 2016). It considers a fixed list of 17 possible POS tags, e.g., noun, verb, adjective, adverb, and preposition. Furthermore, each word has morphological features, such as person, number, and gender. According to UD, depending on the language, there are four different possibilities for the gender of a word; feminine, masculine, neuter, and common (non-neuter). In this paper, we focus on the gender morphological feature of the words.

Gender Bias Analysis Framework. WinoMT (Stanovsky et al., 2019) is the first challenge test set used to evaluate gender bias in MT systems. The test set consists of 3888 sentences; 1826 male sentences, 1822 female sentences, and 240 neutral sentences. It is also distributed with 1584 anti-stereotype sentences, 1584 pro-stereotype sentences, and 720 neutral sentences. Each sentence contains two personal entities where one is a coreferent to a pronoun, and a golden gender is specified for this entity. Three metrics are used for assessment: accuracy (Acc.), which is measured by comparing the translated entity with the golden entity, ΔG and ΔS. ΔG is the difference between the correctly inflected masculine and feminine entities. ΔS is the difference between the inflected genders of the pro-stereotype and anti-stereotype entities. (Saunders and Byrne, 2020) also propose **M:F**, which is the ratio of hypotheses with masculine predictions to those with feminine predictions. ΔS can be skewed in low-accuracy systems; thus, **M:F** would be easier to interpret. Ideally, the absolute values of ΔS and ΔG should be closer to 0, and **M:F** should be closer to 1.

4 Proposed methodology

Previous work (Costa-jussà et al., 2020) has shown that the language-specific multilingual NMT architecture proposed by (Escolano et al., 2021) outperformed the universal shared encoder-decoder architecture (Johnson et al., 2017) on gender bias evaluations. We chose this language-specific architecture as the baseline for all our experiments.

Given parallel data for a set of languages $L = l_1, l_2, ..l_n$ with n languages and data for language pairs, the architecture consists of n encoders and n decoders, each of them specific for a single lan-

[1] https://universaldependencies.org/

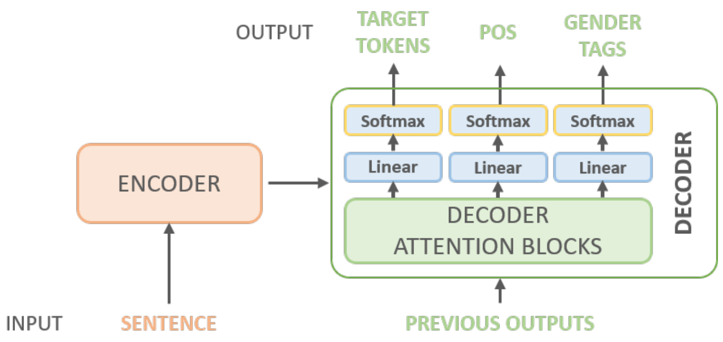


Figure 1: The model architecture, each task has its linear layer, softmax, and loss function

guage. Being l_i encoder used for all translation directions involving l_i as source language and l_i decoder for all directions involving l_i as target language. Using the same encoders and decoders on several translation directions enforces a common intermediate representation for all encoders in the system.

This method relies on parallel data only, without any additional linguistic information. Our proposed modification to this architecture (see Figure 1) adds a new linear projection layers to the decoders for each additional tagging task $tag_k \in T$. Each task focuses on different linguistic aspects that may improve gender representation. It is known that multitasking allows models, by inductive bias (Ruder, 2019), to learn a representation that contains features useful for all the involved tasks, improving its generalization capabilities. For each translation direction, the loss is computed as follows:

$$L(t, y', y) = L_{tr}(y', y) + \sum_{k=1}^{K} L_{tag}(t_k, tag_k(y)) \tag{1}$$

Where y' are the translation logits, y is the reference target sentence, L_{tr} is the cross-entropy loss for the translation task, t is the set of tagging logits for each of the K tagging tasks, L_{tag} is the cross-entropy loss over each tagging task and $tag_k(y)$ is the tagging function over the reference target.

5 Experimental Framework

In this section, we describe the details about the data and model parameters involved in our experiments.

5.1 Data and preprocessing

We have used *Europarl* dataset as training data between all 12 possible language pairs between Spanish, German, English, and French. For each pair, approximately 2 million sentences were available, for a total amount of 24 million. As validation and evaluation data for NMT results, newstest2012 and newstest2013 (Bojar et al., 2013). Only results with English as the source are provided to match the Gender Bias evaluation framework. All data has been preprocessed by applying tokenization, punctuation normalization, and true-casing using standard *Moses* (Koehn et al., 2007) scripts and tokenized at BPE subword level (Sennrich et al., 2016) with 32 thousand steps using subword-nmt framework[2].

Linguistic features have been extracted using the *Stanza* framework (Qi et al., 2020) at word level. For split words, a tag is repeated. For Gender bias evaluation, WinoMT dataset has been used. All data has been preprocessed following the same pipeline depicted for NMT.

5.2 Model Parameters

All models are implemented on *fairseq*'s (Ott et al., 2019) [3] Transformer (Vaswani et al., 2017) with 6 attention layers on both encoder and decoder, 512 embedding size, 8 attention heads and 2048 feed-forward size. Each translation direction has approximately 60 million parameters. POS tag and Gender predictions only account for 8704 and 2048 additional parameters compared to the baseline system. All models have been trained using no further improvement of the validation loss as an early stopping criterion.

[2] https://github.com/rsennrich/subword-nmt

[3] https://github.com/pytorch/fairseq version 0.6

	en-es	en-fr	en-de	es-en	es-fr	es-de	fr-en	fr-es	fr-de	de-en	de-es	de-fr	Avg
Baseline	**29.48**	**29.56**	21.5	**27.38**	**30.25**	19.61	26.03	29.04	18.96	24.04	**24.95**	**25.24**	**25.50**
+POS	29.06	29.27	21.4	27.16	29.79	19.43	25.92	28.89	18.91	**24.24**	24.84	24.99	25.32
+Gender	29.38	**29.56**	**21.81**	27.11	30.05	**19.84**	26.13	29.09	18.99	24.07	24.86	25.16	**25.50**
+POS&Gender	29.12	29.37	21.59	26.92	29.9	19.48	**26.54**	**29.2**	**19.24**	24.23	24.91	24.96	25.45

Table 1: Results in terms of BLEU for different language pairs for baseline, the baseline trained with POS tagging task, the baseline trained with gender tagging task and the baseline trained with aforementioned tasks together.

	Spanish				French				German			
	Acc↑	ΔG↓	ΔS↓	M:F↓	Acc↑	ΔG↓	ΔS↓	M:F↓	Acc↑	ΔG↓	ΔS↓	M:F↓
Baseline	57.7	15.1	18.5	2.85	48.8	23.6	9.5	3.99	62.7	9.9	12.4	2.3
+POS	63.7	7.8	15.5	2.27	51.8	17.5	7.4	3.13	**68.3**	**3.5**	8.8	**1.749**
+Gender	58.2	15.2	**7.2**	2.97	49	21.6	**3.8**	3.52	61.3	8.8	**6.4**	2.06
+POS&Gender	**64.5**	**7.1**	13.3	**2.16**	**54.8**	**14.3**	13.8	**2.8**	65.3	5.5	8	1.95

Table 2: WinoMT Results for the three languages Spanish, French and German

6 Results

In this section, we explore the improvements achieved by multi-task training, whether regarding the general translation accuracy or the gendered results.

Translation results. Table 1 show the translation performance of all proposed configurations. Looking at their average performance, we observe that the addition of tagging tasks does not significantly impact the average performance, with a difference of less than 0.2 between all systems.

When looking at individual directions, we can see that training gender tagging task with NMT in English-to-German has improved. We can argue that German is a higher morphological language than English, and the gender tagging task helps the system inject more knowledge about gender in German, leading to better translation accuracy, up to 0.31 for the English-to-German pair compared to the baseline. Training more than one task seems to be beneficial when the translation is between high morphological language pairs like French-to-Spanish and French-to-German where French, German, and Spanish are all gendered high morphological languages. French-to-English seems to also benefit from the POS and Gender tagging tasks together.

WinoMT results. WinoMT helps us investigate how the gender-biased entities and professions translated. The framework investigates the translations when English is translated to higher morphological languages; therefore, we show the WinoMT results from English to Spanish, German and French.

Spanish seems to benefit from the POS and Gender tagging tasks together, where the accuracy increased by 6.8 over the baseline with a lower difference between the correct translated masculine entities and the correct translated feminine entities; of ΔG 7.1. This is assured by the lower ratio of male vs. female predictions (**M:F**) of 2.16. French also has better accuracy having both tasks trained together, with 54.8 accuracy, which increases by six over the baseline. The ΔG is also lower in this case, with a value of 14.3. In both languages, the difference between stereotyped and non-stereotyped translations did not improve. The improvements are more related to the general accuracy.

Multitask training of gender seems to impact the stereotyped translations in the three languages; however, the general accuracy was not impacted that much by training the gender tagging task.

POS tagging also appears to help disambiguation of gender from English to German, giving higher gender accuracy reaching 68.3 with a low difference of male and female correct predictions; ΔG of 3.5 and **M:F** of 1.749.

7 Conclusions

In this paper, we have proposed and analyzed the use of multi-task learning in multilingual NMT. Learning linguistic tagging simultaneously as multilingual helps mitigate gender bias while maintaining the average translation performance over the tested languages. More than the methodology that we are proposing, which is simple and effective, we would like to encourage the community to evaluate their methodologies not only in terms of translation quality, but also in terms of social bias mitigation.

Acknowledgements

This work is supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 947657).

References

Christine Basta, Marta Ruiz Costa-Jussà, and José Adrián Rodríguez Fonollosa. 2020. Towards mitigating gender bias in a decoder-based neural machine translation model by adding contextual information. In *Proceedings of the The Fourth Widening Natural Language Processing Workshop*, pages 99–102. Association for Computational Linguistics.

G. Blackwood, Miguel Ballesteros, and T. Ward. 2018. Multilingual neural machine translation with task-specific attention. *ArXiv*, abs/1806.03280.

Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (technology) is power: A critical survey of "bias" in NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5454–5476, Online.

Ondrej Bojar, Christian Buck, Chris Callison-Burch, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Herve Saint-Amand, Radu Soricut, and Lucia Specia, editors. 2013. *Proceedings of the Eighth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Sofia, Bulgaria.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Marta R. Costa-jussà and Adrià de Jorge. 2020. Fine-tuning neural machine translation on gender-balanced datasets. In *Proceedings of the Second Workshop on Gender Bias in Natural Language Processing*, pages 26–34, Barcelona, Spain (Online). Association for Computational Linguistics.

Marta R. Costa-jussà, Carlos Escolano, Christine Basta, Javier Ferrando, Roser Batlle, and Ksenia Kharitonova. 2020. Gender bias in multilingual neural machine translation: The architecture matters. *CoRR*, abs/2012.13176.

Marta R. Costa-jussà. 2019. An analysis of gender bias studies in natural language processing. *Nature Machine Intelligence*, 1(11):495–496.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 72–78, Vancouver, Canada. Association for Computational Linguistics.

Carlos Escolano, Marta R. Costa-jussà, José A. R. Fonollosa, and Mikel Artetxe. 2021. Multilingual machine translation: Closing the gap between shared and language-specific encoder-decoders. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 944–948, Online. Association for Computational Linguistics.

Joel Escudé Font and Marta R. Costa-jussà. 2019. Equalizing gender bias in neural machine translation with word embeddings techniques. In *Proceedings of the First Workshop on Gender Bias in Natural Language Processing*, pages 147–154, Florence, Italy. Association for Computational Linguistics.

Orhan Firat, Baskaran Sankaran, Yaser Al-onaizan, Fatos T. Yarman Vural, and Kyunghyun Cho. 2016. Zero-resource translation with multi-lingual neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 268–277, Austin, Texas. Association for Computational Linguistics.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Trans. Assoc. Comput. Linguistics*, 5:339–351.

Eliyahu Kiperwasser and Miguel Ballesteros. 2018. Scheduled multi-task learning: From syntax to translation. *Transactions of the Association for Computational Linguistics*, 6:225–240.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 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, Prague, Czech Republic. Association for Computational Linguistics.

Yichao Lu, Phillip Keung, Faisal Ladhak, Vikas Bhardwaj, Shaonan Zhang, and Jason Sun. 2018. A neural interlingua for multilingual machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 84–92, Brussels, Belgium. Association for Computational Linguistics.

Amit Moryossef, Roee Aharoni, and Yoav Goldberg. 2019. Filling gender & number gaps in neural machine translation with black-box context injection. In *Proceedings of the First Workshop on Gender Bias in Natural Language Processing*, pages 49–54.

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.

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*.

Marcelo Prates, Pedro Avelar, and Luís Lamb. 2020. Assessing gender bias in machine translation: a case study with google translate. *Neural Computing and Applications*, 32.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Sebastian Ruder. 2019. *Neural Transfer Learning for Natural Language Processing*. Ph.D. thesis, National University of Ireland, Galway.

Danielle Saunders and Bill Byrne. 2020. Reducing gender bias in neural machine translation as a domain adaptation problem.

Beatrice Savoldi, Marco Gaido, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2021. Gender bias in machine translation.

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, Florence, Italy. Association for Computational Linguistics.

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.

Artūrs Stafanovičs, Toms Bergmanis, and Mārcis Pinnis. 2020. Mitigating gender bias in machine translation with target gender annotations.

Gabriel Stanovsky, Noah Smith, and Luke Zettlemoyer. 2019. Evaluating gender bias in machine translation. pages 1679–1684.

Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, and William Yang Wang. 2019. Mitigating gender bias in natural language processing: Literature review. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1630–1640, Florence, Italy. Association for Computational Linguistics.

Eva Vanmassenhove, Christian Hardmeier, and Andy Way. 2018. Getting gender right in neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3003–3008, Brussels, Belgium. 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.

Poorya Zaremoodi and Gholamreza Haffari. 2018. Neural machine translation for bilingually scarce scenarios: a deep multi-task learning approach. 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 1356–1365, New Orleans, Louisiana. Association for Computational Linguistics.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. 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 15–20, New Orleans, Louisiana. Association for Computational Linguistics.

Changfeng Zhu, Heng Yu, Shanbo Cheng, and Weihua Luo. 2020. Language-aware interlingua for multilingual neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1650–1655, Online. Association for Computational Linguistics.

Small Batch Sizes Improve Training of Low-Resource Neural MT

Àlex R. Atrio[1,2] and Andrei Popescu-Belis[1,2]

[1]HEIG-VD / HES-SO, Yverdon-les-Bains, Switzerland
[2]EPFL, Lausanne, Switzerland

{alejandro.ramirezatrio, andrei.popescu-belis}@heig-vd.ch

Abstract

We study the role of an essential hyper-parameter that governs the training of Transformers for neural machine translation in a low-resource setting: the batch size. Using theoretical insights and experimental evidence, we argue against the widespread belief that batch size should be set as large as allowed by the memory of the GPUs. We show that in a low-resource setting, a smaller batch size leads to higher scores in a shorter training time, and argue that this is due to better regularization of the gradients during training.

1 Introduction

Training Transformers for low-resource neural machine translation (NMT), i.e. when only small parallel corpora are available, raises the challenge of finding optimal hyper-parameters. While several fixed configurations of the Transformer (Vaswani et al., 2017) have been empirically validated by the community, such as 'Base' or 'Big', the settings of many other hyper-parameters rely on tips from practitioners. However, these values are not always suitable to low-resource settings, and systematic studies in these settings are rare (Araabi and Monz, 2020; Van Biljon et al., 2020).

In this paper, we show that the best values of a hyper-parameter that is essential for training, namely *batch size*, differ in low-resource settings from those commonly accepted when larger data sets are available. We analyze the role of small batch sizes, inspired by studies in computer vision (Keskar et al., 2016), and then pinpoint empirically the optimal trade-off between a high batch size (for efficiency) and a small one (for regularization). Although large batch sizes were found to lead to higher-quality models in experiments with high-resource NMT (Popel and Bojar, 2018; Xu et al., 2020), we show here that smaller batch sizes can outperform the latter, likely due to a regularizing effect in the gradient update. Moreover, we show that this finding is invariant to changes in tokenization methods.

The paper is organized as follows. In Section 2, we discuss batch size from a machine learning perspective, showing why smaller values of batch size may act as regularizers. Then, in Section 3, we review studies of hyper-parameters in NMT. In Section 4, we present the parameters of our Transformer and the data from the WMT 2020 Low-resource task (Fraser, 2020) and other sources that we use in our experiments. In Section 5, we provide empirical evidence that smaller batch sizes are preferable in low-resource settings.

2 ML Perspective on Batch Size

Machine learning theory argues that performing back-propagation with large batch sizes leads to better optimization, because the estimates of the gradients are more accurate. Conversely, using small batches during training leads to noisier gradient estimations, i.e. with a larger variance in comparison to the gradient computed over the entire training set. Still, one advantage of small batch sizes is that they are more likely to make parameters converge towards flatter minima of the loss (Goodfellow et al., 2016, Chapter 8.1.3), as explained below. Such flatter minima have better generalization capacities, i.e. they maintain performance when presented with a new test set.

Keskar et al. (2016) define a *flat minimizer* – as opposed to a *sharp* one – as a point in the parameter space that is a local minimum of the loss function, and where this function varies slowly in a relatively large neighborhood. Keskar et al. (2016) point to the following *generalization gap*: training with large batch sizes tends to converge towards sharp minimizers, which offer poorer generalization capacities. Conversely, *small batch sizes allow convergence towards flat minimizers*, which are likely to generalize better. Thus, smaller batch sizes have *exploration abilities*: the search is more likely to exit the basins of sharp minimizers, and to tend towards flat minimizers, from where noise will not cause them to exit.

Since a sharp minimizer requires high precision to be described, unlike a flat one, the more noise there is in the gradient, the more unlikely it is that the parameters will converge towards a sharp minimizer. This is precisely the contribution of a smaller batch size: introduce noise in the gradient estimation. According to this theoretical view, above a certain threshold of the batch size, the generalization capacities of a model deteriorate. The threshold depends on several hyper-parameters, including the batch size. Its role has not been fully settled yet, with observations and conclusions varying widely across studies (Dinh et al., 2017; Hoffer et al., 2017; Goyal et al., 2017; Li et al., 2017; Kawaguchi et al., 2017). Moreover, these studies are on image data sets, with fully connected or with convolutional NNs, which differ substantially from NMT settings.

3 The Role of Batch Size in Neural MT

Several recent studies in NMT have considered batch size among other hyper-parameters, but they have either been in high-resource settings (Popel and Bojar, 2018; Xu et al., 2020) or have given only marginal attention to batch size (Sennrich and Zhang, 2019; Araabi and Monz, 2020).

Popel and Bojar (2018) reported that BLEU scores increased with batch size (including when using more GPUs) in a Transformer-based NMT system, although with diminishing returns, recommending in particular that "batch size should be set as high as possible". Their experiments were performed using mainly two datasets, with respectively 58M and 15M sentence pairs. It thus remains an open question whether their findings regarding batch size also apply when much less training data is available.

Sennrich and Zhang (2019) experimented with a recurrent network in a low-resource setting and found that smaller batch sizes were beneficial, along with other forms of regularization. They experimented with two batch sizes of 4,000 and 1,000 tokens, and observed improvements with the latter of 0.30 and 0.04 BLEU points on data sets with 5k and 160k sentence pairs, respectively. It is difficult to predict from these results what the optimal batch size is for Transformer-based NMT.

Araabi and Monz (2020) studied the role of 15 hyper-parameters of the Transformer, with several sizes of low-resource datasets. For the largest training sizes tested (80k and 165k sentence pairs), larger batch sizes improved performance, with respectively 8,192 and 12,288 versus 4,096 for the other sizes. For lower training sizes, smaller batch sizes did not improve performance, which the authors explain by Transformer's need for larger batches. In our view, an alternative explanation is the order of optimization of the hyper-parameters (a grid search in which they optimize one hyper-parameter at a time): batch size is #12 out of 15, so by the time several sizes are compared, regularization has already been introduced in the model by dropouts on words, activation, and layers. Late optimization of batch size, of warmup steps (#14) or of learning rate (#15) cannot properly determine their regularizing effects.

Xu et al. (2020) proposed to compute gradients while accumulating minibatches, and observed that increasing batch size stabilizes gradient direction up to a certain point, after which it starts to fluctuate. They used this criterion to dynamically adjust batch sizes while training. In their experiments with large training sets (4.5M and 36M sentence pairs), their average batch size was around 26k on two GPUs, and never lower than 7k. Their observations on the gradient direction as more minibatches are accumulated are consistent with the findings of Popel and Bojar (2018) who see diminishing returns when increasing batch size.

4 Datasets and Systems

We train NMT systems with two low-resource parallel corpora, listed in the first two lines of Table 1: the Upper Sorbian (HSB) to German (DE) training data of the WMT 2020 Low-Resource Translation Task (Fraser, 2020) and a low-size excerpt of the German to English News Commentary v13 (Bojar et al., 2018), from which we randomly sampled 60k parallel lines. For the HSB-DE models, we also use the development and test sets provided by the WMT 2020 and 2021 Low-Resource Translation Task (Libovický and Fraser, 2021), each consisting of 2k sentences, and for DE-EN we sample a development set and a test set from the original corpus, with 2k sentences each as well. We apply a com-

Dataset	Lang.	Orig.	Filt.	Δ%
WMT20 Low-res.	HSB-DE	60k	59.8k	0.29
News Comm. v13	DE-EN	60k	59.9k	0.20
Sorbian Institute	HSB	339k	339k	0.00
Witaj	HSB	222k	220k	0.84
Web	HSB	134k	121k	9.98
Europarl v8	DE	2.2M	2.2M	0.79
News Comm. v15	DE	422k	411k	2.58
JW300	DE	2.3M	2.2M	4.44
Europarl v3	DE	790k	785k	0.69
Europarl v3	EN	790k	782k	1.07

Table 1: Numbers of lines in the original and filtered corpora used in our experiments. HSB stands for Upper Sorbian and Δ% for the proportion of lines filtered out. The only *parallel* corpora used for training NMT are the first two ones; the other corpora are only used to train the SentencePiece model.

mon filtering process for all data used: we delete from all our data the sentences that are not between 2 and 300 words long, with resulting numbers of lines shown in Table 1.

We build subword vocabularies using the Unigram LM model (Sennrich et al., 2016; Kudo, 2018) as implemented in SentencePiece[1], with the monolingual corpora from Table 1. We train a shared model for HSB-DE with a vocabulary of 32k pieces, character coverage of 0.98, *nbest* = 1 and *alpha* = 0. The HSB data adds up to 740k sentences, and we sample 680k sentences from three DE corpora, and add them to the 60k sentences from the DE side of the parallel HSB-DE corpus. To train the SentencePiece model for the DE-EN, for comparison purposes, we treat German as a low-resource language, and sample 680k lines of English and German from Europarl v3 (Tiedemann, 2012), which we combine respectively with the 60k lines extracted from the DE-EN parallel corpus.

We use the Transformer-Base (Vaswani et al., 2017) in the implementation provided by OpenNMT (Klein et al., 2017, 2020), with the parameters given in Appendix A. Unless otherwise specified, we follow OpenNMT-py's recommended values for the hyper-parameters.[2]

When using several GPUs with gradient accumulation, each GPU processes several batches, which are then accumulated across all GPUs and used to update the model at each step. Therefore, the *effective batch size* is $B \times G \times A$, where B is the individual batch size, G is the number of GPUs

[1] https://github.com/google/sentencepiece

[2] https://opennmt.net/OpenNMT-py/examples/Translation.html

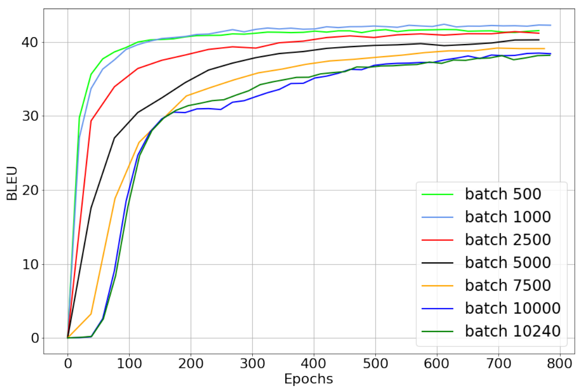


Figure 1: BLEU scores on the test set for HSB-DE models trained with different batch sizes.

and A the number of accumulated batches, and differs from the `batch_size` hyper-parameter B. We train all models on two GeForce RTX 1080Ti GPUs with 11 GB of memory each and accumulate gradients over two minibatches ($A = 2$), following OpenNMT-py's recommendation. Therefore, the `batch_size` parameter is not our effective batch size, which is four times larger. Throughout this work, we will refer to batch size B as the `batch_size` parameter, and report true *epochs*, which we define as computed with the effective batch size as $S \times B_{\mathit{eff}}/N$, for S training steps, B_{eff} effective batch size, and N number of source tokens in the training set.

Following OpenNMT-py's recommendations, we set the Adam hyper-parameters at $\beta_1 = 0.9$, $\beta_2 = 0.998$, $\epsilon = 10^{-8}$ and apply at each step a scaling factor of two to Noam's learning rate schedule, setting warmup steps to 8k. Translations are generated with a beam width of seven, with an ensemble of the last four saved checkpoints. We report BLEU scores (Papineni et al., 2002) obtained with SacreBLEU (Post, 2018) on detokenized text.

5 Experimental Results

To study the impact of batch sizes in a low-resource setting, we train various HSB-DE and DE-EN models for 700 epochs with the following batch sizes: 100, 250, 500, 1,000, 2,500, 5,000, 7,500, 10,000, and 10,240 (this is the largest one that fits in our GPU memory).

5.1 NMT Performance

NMT performance on the HSB-DE test set throughout the training is shown in Figure 1, with BLEU scores depending on the number of epochs. The evolution depending on real training time (wall

Batch	HSB-DE								DE-EN							
Size	dev				test				dev				test			
	Xent	BLEU	chrF	TER	Xent	BLEU	chrF	TER	Xent	BLEU	chrF	TER	Xent	BLEU	chrF	TER
500	0.03	48.12	71.13	37.35	0.03	41.53	67.34	43.84	0.11	37.35	58.04	54.54	0.11	37.72	58.35	54.60
1,000	**0.02**	**49.23**	**72.07**	**36.35**	**0.02**	**42.26**	**67.93**	**43.16**	0.05	**38.03**	**59.39**	**52.91**	0.05	**38.67**	**59.68**	**52.71**
2,500	0.03	48.28	71.63	37.02	0.03	41.18	67.36	44.02	**0.04**	33.83	56.70	56.27	**0.04**	35.51	57.76	55.47
5,000	0.03	46.99	70.74	38.05	0.03	40.28	66.62	45.24	0.05	32.47	55.20	57.88	0.05	33.97	56.16	57.08
7,500	0.03	46.05	70.29	38.87	0.03	39.10	65.94	46.18	0.05	32.67	55.99	57.67	0.05	33.80	56.72	57.21
10,000	0.04	44.61	69.19	40.00	0.04	38.41	65.67	46.45	0.05	31.84	55.20	58.35	0.05	33.50	56.14	57.63
10,240	0.04	45.59	70.12	39.26	0.04	38.19	65.39	46.79	0.06	31.49	55.00	58.65	0.06	33.03	55.78	58.07

Table 2: Loss and scores for models trained for 700 epochs with various batch sizes for HSB-DE and DE-EN directions. All differences in BLEU on the dev and test sets are statistically significant at the 95% level, except for the pairs in similar colors.

time) is similar in terms of rankings. Thus, the following analysis holds whether we train the models for the same amount of epochs or of hours.

The final scores on the development and test sets are given in Table 2, sorted by batch sizes. We provide first the actual loss of the model ('Xent' for cross-entropy), and then three typical NMT scores: BLEU (Papineni et al., 2002), chrF (Popović, 2015) and Translation Error Rate (Snover et al., 2006). The 100 and 250 batch size models did not reach BLEU scores significantly above zero, and are not included among the results in the table.

We test the statistical significance of the differences between each score and the others, with 95% confidence, using the paired bootstrap resampling tool from SacreBLEU (Post, 2018).[3] All differences between higher and lower BLEU scores are statistically significant, except the pairs highlighted in similar colors in Table 2.[4] The best NMT scores, which are always obtained with a batch size of 1,000, are significantly higher than all the other ones, including those obtained with the largest possible batch sizes for our GPU (10,000 or 10,240). We thus select two values for further experiments: a batch size of 1,000 as our highest-scoring model, and one of 10,000 as the maximum allowed by our GPU memory. A simple ratio of 10 holds between the two values.

These empirical results are contrary to those from Popel and Bojar (2018), who observe that increasing the batch size for Transformer-Base produces higher scores, although with diminishing returns after a certain threshold. We hypothesize that the main explanation is the difference between the amounts of training data: in our low-resource setting, we use 60k sentences, while Popel and Bojar (2018) use 57M sentences. Our findings are consistent with those of Keskar et al. (2016), who also observe that the optimal batch size is at the lower end of the range, on a computer vision task with convolutional and fully-connected NNs.

5.2 Asymptotic Performance

An alternative explanation for the previous results is that the learning rate is too small for the larger batch sizes, which require more time to converge. To test whether the differences observed above between small and large batch sizes depend on the actual training time, we continue training the 1,000 and 10,000 batch size models for HSB-DE and DE-EN for twice as many epochs as above (1400). The BLEU scores and their increases with respect to training for 700 epochs are given in Table 3. The performance gap (from +3.85 to +3.25 BLEU) between small and large batch sizes is not overturned by training the models for much longer.

The scores from our best system (1,000 batch size, 42.81 BLEU on the test set) are similar to scores obtained by *baselines* of the five highest-scoring teams at the WMT20 Low-resource shared task on HSB-DE (Fraser, 2020). While the scores of Scherrer et al. (2020) and Li et al. (2020) are not comparable due to a different architecture or the use of unsupervised pre-training, the baseline scores of Knowles et al. (2020), Libovický et al. (2020) and Kvapilíková et al. (2020) are respectively 44.1, 43.4, and 38.7. The first one is higher than our best BLEU by 1.29, likely due to the use of 43M lines of CS and DE data for the subword vocabulary, vs. 700k in our case.

[3]github.com/mjpost/sacrebleu with the signature nrefs:1|bs:1000|seed:12345|case:mixed|eff:no|tok:13a|smooth:exp|version:2.0.0.

[4]The difference in BLEU between the following pairs is not significant. For HSB-DE, 2,500 vs. 500, and 10,240 vs. 10,000, on the test set; and 2,500 vs. 500, and 7,500 vs. 10,240 on the dev set. For DE-EN these are 7,500 vs. 5,000, and 10,000 vs. 7,500, on the test set; and 1,000 vs. 500, 5,000 vs. 7,500, and 10,000 vs 10,240, on the dev set.

Batch size	HSB-DE		DE-EN	
	dev	test	dev	test
1,000	49.52 (*+0.29*)	42.81 (*+0.55*)	38.67 (*+0.64*)	39.24 (*+0.57*)
10,000	46.44 (*+1.83*)	39.56 (*+1.15*)	33.19 (*+1.35*)	34.42 (*+0.92*)

Table 3: BLEU scores for models trained for 1,400 epochs. The scores for 1,000 are significantly higher (at 95%) than those for 10,000. In parenthesis, the absolute difference with BLEU scores after 700 epochs.

5.3 Invariance with respect to Vocabulary

We additionally perform two comparisons that show that the above results hold regardless of the tokenizer and the vocabulary size. First, we test whether the score difference is preserved with an unshared SentencePiece vocabulary, i.e. when not sharing the source (HSB and DE) and the target (DE and EN) vocabularies.

Second, we train two NMT models for HSB-DE using a Byte Pair Encoding (BPE) vocabulary (Sennrich et al., 2016), which we generate using the `learn_bpe.py` tool from OpenNMT-py, with 32k merge operations and the remaining parameters at default values. Table 4 shows BLEU scores on the development sets for batch sizes of 1,000 and 10,000. The previously observed differences in score between the batch sizes still hold, and we see that a shared SentencePiece vocabulary leads to a better NMT system than an unshared or a BPE one.

Batch size	SP unshared		BPE
	HSB-DE	DE-EN	HSB-DE
1,000	46.80 (*-2.43*)	35.90 (*-2.13*)	46.21 (*-3.02*)
10,000	41.99 (*-2.62*)	30.09 (*-1.75*)	43.35 (*-1.26*)

Table 4: BLEU scores on the dev set for HSB-DE and DE-EN models trained with SentencePiece (SP) vocabularies not shared between source and target (left) and BPE subwords (right). The scores for 1,000 are significantly higher (at 95%) than those for 10,000. In parenthesis, the difference with BLEU scores obtained with the SP shared vocabulary.

6 Conclusion and Future Work

In this work, we have shown that insights from computer vision on the regularizing effect of small batch sizes are also applicable to NMT. Our results, focused on a low-resource setting, challenge those of previous NMT studies with large amounts of training data, and the general belief that batch sizes should be as large as they fit in the GPU memory. We have shown that training with small batch sizes leads to models that generalize better, and found the optimal batch size below which performance degrades.

Future work should explore how the learning rate must be adjusted depending on the batch size, and whether a dynamically scheduled combination of batch size and learning rate can provide an even better regularizer. For instance, it should be tested if dynamic batch sizes as proposed by Xu et al. (2020) can also improve performance in a low-resource setting, with batch size thresholds changed to measure an optimal level of noise.

Acknowledgments

We thank for their financial support the Swiss National Science Foundation (grant n. 175693 for the DOMAT project: On-demand Knowledge for Document-level Machine Translation) and Armasuisse (FamilyMT project). We especially thank Dr. Ljiljana Dolamic (Armasuisse) for her support in the FamilyMT project. We are also grateful to the anonymous reviewers and to Giorgos Vernikos for their helpful suggestions.

References

Ali Araabi and Christof Monz. 2020. Optimizing Transformer for low-resource neural machine translation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3429–3435, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels. Association for Computational Linguistics.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. 2017. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pages 1019–1028. PMLR.

Alexander Fraser. 2020. Findings of the WMT 2020 shared tasks in unsupervised MT and very low resource supervised MT. In *Proceedings of the Fifth Conference on Machine Translation*, pages 765–771, Online. Association for Computational Linguistics.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, large minibatch SGD: Training Imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*.

Elad Hoffer, Itay Hubara, and Daniel Soudry. 2017. Train longer, generalize better: Closing the generalization gap in large batch training of neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1729–1739, Red Hook, NY, USA. Curran Associates Inc.

Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. 2017. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Guillaume Klein, François Hernandez, Vincent Nguyen, and Jean Senellart. 2020. The OpenNMT neural machine translation toolkit: 2020 edition. In *Proceedings of the 14th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 102–109, Virtual. Association for Machine Translation in the Americas.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for NMT. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72. Association for Computational Linguistics.

Rebecca Knowles, Samuel Larkin, Darlene Stewart, and Patrick Littell. 2020. NRC systems for low resource German-Upper Sorbian machine translation 2020: Transfer learning with lexical modifications. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1112–1122, Online. Association for Computational Linguistics.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75.

Ivana Kvapilíková, Tom Kocmi, and Ondřej Bojar. 2020. CUNI systems for the unsupervised and very low resource translation task in WMT20. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1123–1128, Online. Association for Computational Linguistics.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2017. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*.

Zuchao Li, Hai Zhao, Rui Wang, Kehai Chen, Masao Utiyama, and Eiichiro Sumita. 2020. SJTU-NICT's supervised and unsupervised neural machine translation systems for the WMT20 news translation task. In *Proceedings of the Fifth Conference on Machine Translation*, pages 218–229, Online. Association for Computational Linguistics.

Jindřich Libovický, Viktor Hangya, Helmut Schmid, and Alexander Fraser. 2020. The LMU Munich system for the WMT20 very low resource supervised MT task. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1104–1111, Online. Association for Computational Linguistics.

Jindřich Libovický and Alexander Fraser. 2021. Findings of the WMT 2021 shared tasks in unsupervised MT and very low resource supervised MT. In *Proceedings of the Sixth Conference on Machine Translation*, Online. 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 of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Martin Popel and Ondřej Bojar. 2018. Training tips for the Transformer model. *The Prague Bulletin of Mathematical Linguistics*, 110(1):43–70.

Maja Popović. 2015. chrF: character n-gram f-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. 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.

Yves Scherrer, Stig-Arne Grönroos, and Sami Virpioja. 2020. The University of Helsinki and Aalto university submissions to the WMT 2020 news and low-resource translation tasks. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1129–1138, Online. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. NMT 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.

Rico Sennrich and Biao Zhang. 2019. Revisiting low-resource neural machine translation: A case study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 211–221, Florence, Italy. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotations. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA 2006)*.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Elan Van Biljon, Arnu Pretorius, and Julia Kreutzer. 2020. On optimal Transformer depth for low-resource language translation. *arXiv preprint arXiv:2004.04418*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008.

Hongfei Xu, Josef van Genabith, Deyi Xiong, and Qiuhui Liu. 2020. Dynamically adjusting Transformer batch size by monitoring gradient direction change. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3519–3524, Online. Association for Computational Linguistics.

A Appendix

The hyper-parameters used to train our models are the following ones:

```
src_words_min_frequency: 2
tgt_words_min_frequency: 2
valid_batch_size: 200
max_generator_batches: 2
optim: adam
learning_rate: 2.0
adam_beta2: 0.998
decay_method: noam
accum_count: 2
warmup_steps: 8000
label_smoothing: 0.1
max_grad_norm: 0
param_init: 0
param_init_glorot: true
normalization: tokens
encoder_type: transformer
decoder_type: transformer
position_encoding: true
layers: 6
heads: 8
rnn_size: 512
word_vec_size: 512
transformer_ff: 2048
dropout: 0.1
batch_size: 1000
batch_type: tokens
```

lakṣyārtha (Indicated Meaning) of Śabdavyāpāra (Function of a Word) framework from kāvyaśāstra (The Science of Literary Studies) in Samskṛtam :Its application to Literary Machine Translation and other NLP tasks

Sripathi Sripada **Anupama Ryali** **Raghuram Sheshadri**
School of Vedic Sciences, MIT-ADT University, India
{sripathi.sripada, anupamaskt, raghuys}@gmail.com

Abstract

A key challenge in Literary Machine Translation is that the meaning of a sentence can be different from the sum of meanings of all the words it possesses. This poses the problem of requiring large amounts of consistently labelled training data across a variety of unsages and languages. In this paper, we propose that we can economically train machine translation models to identify and paraphrase such sentences by leveraging the language independent framework of Śabdavyāpāra (Function of a Word), from Literary Sciences in Saṃskṛtam, and its definition of lakṣyārtha ('Indicated' meaning). An Indicated meaning exists where there is incompatibility among the literal meanings of the words in a sentence (irrespective of language). The framework defines seven categories of Indicated meaning and their characteristics. As a pilot, we identified 300 such sentences from literary and regular usage, labelled them and trained a 2d Convolutional Neural Network to categorise a sentence based on the category of Indicated meaning and finetuned a T5 to paraphrase them. We then used these paraphrased sentences as input into Google Translate and compared this with Google Translate's translation before paraphrasing using BLEU scores against an expected reference translation. The BLEU scores improved significantly with the paraphrasing by the T5 trained on Indicated meaning sentences.

Keywords: Indicated meaning, Literary Machine Translation, language independent, T5[1], Convolutional Neural Network, paraphrase.

1 Introduction

Consider a sentence from Rabindranath Tagore's Gitanjali "Drunk by the joy of singing, I forget myself", and its translation to various languages using Google Translate[2]. Refer to Table 1 for sample translations

Language	**Google Translate's Translation**
Hindi	गाने के आनंद के नशे में धुत मैं खुद को भूल जाता हूँ gaane ke aanand ke nashe mein dhut main khud ko bhool jaata hoon
Bengali	গান গাওয়ার আনন্দে মাতাল হয়ে আমি নিজেকে ভুলে যাই Gāna gā'ōẏāra ānandē mātāla haẏē āmi nijēkē bhulē yā'i
Kannada	ಹಾಡುವ ಸಂತೋಷದಿಂದ ಕುಡಿದು, ನಾನು ನನ್ನನ್ನು ಮರೆತಿದ್ದೇನೆ Hāḍuva santōṣadinda kuḍidu, nānu nannannu maretiddēne
Telugu	పాడిన ఆనందంతో త్రాగి, నన్ను నేను మర్చిపోతున్నాను Pāḍina ānandantō trāgi, nannu nēnu marcipōtunnānu
Italian	Ubriaco dalla gioia di cantare, dimentico me stesso
German	Betrunken von der Freude am Singen vergesse ich mich selbst

Table 1: Sample translations by Google Translate of the example sentence "Drunk by the joy of singing I forget myself"

For the example above, while in some languages the usage could be appropriate, the translation is not very clear in quite a few languages. On the other hand, if we paraphrase the original sentence to "Overjoyed by singing, I forget myself", then

[1] T5 is Google's state of the art text to text NLP model. T5 stands for Text-To-Text Transfer Transformer

[2] Google's publicly available translation engine at https://translate.google.co.in

Google Translate's translation is more consistent across multiple languages. Refer to Table 2 for sample translations of the paraphrased sentence

Language	Google Translate's Translation
Hindi	गाते-गाते खुशी से झूम उठे मैं खुद को भूल गया gaate-gaate khushee se jhoom uthe main khud ko bhool gaya
Bengali	গান গেয়ে আনন্দিত, আমি নিজেকে ভুলে যাই Gāna gēẏē ānandita, āmi nijēkē bhulē yā'i
Kannada	ಹಾಡುವ ಮೂಲಕ ಹರ್ಷಗೊಂಡ ನಾನು ನನ್ನನ್ನೇ ಮರೆತಿದ್ದೇನೆ Hāḍuva mūlaka harṣagoṇḍa nānu nannannē maretiddēne
Telugu	పాడటం ద్వారా చాలా సంతోషించాను, నేను నన్ను మరచిపోయాను Pāḍaṭaṁ dvārā cālā santōṣiñcānu, nēnu nannu maracipōyānu
Italian	Felicissimo di cantare, mi dimentico di me stesso
German	Überglücklich vom Singen vergesse ich mich selbst

Table 2 : Sample translations by Google Translate of the paraphrased sentence "Overjoyed by singing, I forget myself"

There are many such sentences across literary works where the sum of meanings of all the words in a given sentence, does not necessarily provide the meaning of the sentence. In all such cases, an appropriate paraphrasing should make machine translation more accurate. To train machine learning models for paraphrasing of such sentences before translation, we are faced with the challenge of creating large datasets for training across different types of usages, figures or speech etc., and across multiple languages.

Recent related works: Recent research in the applying machine translation models to literary works is broadly focused on:

- training the models to identify and paraphrase metaphors to their literal meanings (Jerry Lui, 2020) (Rui Mao, 2018) leveraging word embeddings
- modifications to existing machine translation models for classification of consistency, pronoun resolution, and tone/register error types to consider context of previous sentences or even the whole story, to improve quality of literary machine translation (Matusov, 2019)
- the role of referential cohesion to improve Literary Machine Translation (Rob Voigt, 2012)

However, to our knowledge, there is lack of a holistic approach that encompasses a variety of the challenges presented in Literary Machine Translation in a manner consistent across languages.

Our approach: To overcome this challenge, we seek inspiration from kāvyaśāstra, the Science of Literary Works / Poetics, in Saṃskṛtam. Various texts in Saṃskṛtam in this domain, provide comprehensive and lucid frameworks to understand literary works. A variety of concepts discussed in these texts are language independent as well. Of many such concepts, kāvyaśāstra lays much importance to a word and its meaning. It emphasises that a word and its meaning depend on the speaker, the listener, and the tone[3]. At times, it is understood with the context too. This framework of understanding the meaning is called Śabdavyāpāra (as explained in kāvyaprakāśa) and it categorises the word and its meaning broadly into three types[4], namely,

- vācakaḥ (वाचकः - Expressive) word with vācyārthaḥ (वाच्यार्थः Expressed) meaning or literal / direct / primary meaning
- lākṣaṇikaḥ (लाक्षणिकः - Indicative) word with lakṣyārthaḥ (लक्ष्यार्थः - Indicated meaning)
- vyañjakaḥ (व्यञ्जकः - Suggestive) word with vyaṅgyārthaḥ (व्यङ्ग्यार्थः - Suggested) meaning.

The Indicated meaning from the above framework provides a very fundamental categorisation of words which covers a variety of

[3] वक्तृ-बोद्धव्य-काकूनां सम्बन्धः। vaktṛ-boddhavya-kākūnāṃ sambandhaḥ. The relationship between speaker, listener, tone.

[4] स्याद्वाचको लाक्षणिकः शब्दोत्र व्यञ्जकस्त्रिधा । syādvācako lākṣaṇikaḥ śabdotra vyañjakastridhā . The words are of 3 types – Expressive, Indicative and Suggestive

figurative usages, metaphors, referential cohesion and other characteristics. Hence, it provides a more holistic approach, in comparison to techniques focusing on metaphors and figures of speech, to solving some of the key problems in Literary Machine Translation. We explain the Indicated meaning, its various types and their characteristics along with examples in section 2.

The Hypothesis: Our hypothesis is that if we train a state of the art NLP model to paraphrase based on the Śabdavyāpāra framework and then use a state of the art Machine Translation model to translate, the translation of Literary works across languages will be much more meaningful, and consistent. Moreover, since the framework is language independent and has a very structured definition of the various types of Indicated meaning and their characteristics, we should be able to achieve a very efficient training with smaller datasets and consistently across languages.

We adopted a novel approach, based on Śabdavyāpāra framework's definition and characteristics of the Indicated meaning, to

a) Train a 2d Convolutional Neural Network (CNN2d) to Identify the existence of an Indicated meaning, in a given sentence
b) If Indicated meaning is present, then train a CNN2d to categorise the sentence based on the type of Indicated meaning, as per the framework
c) Leverage the characteristics of the various types of Indicated meaning defined in the framework to finetune a Google T5 (Transformer NLP model) to paraphrase the sentence by elaborating the Indicated meaning such that the paraphrased sentence can be translated consistently by a model like Google Translate.

To do an initial validation of our hypothesis we created a dataset of 300 sentences from literary works, Śāstra works and common usage[5]. In Section 3, we describe the solution we adopted including the models we trained along with the results we achieved in our pilot. In section 4, we conclude and highlight the other use cases of NLP where the identification and paraphrasing of Indicated meaning can be applicable. In Appendices we provide some examples of the seven categories of Indicated meaning.

2 Śabdavyāpāra (Function of a Word) and lakṣyārtha (Indicated meaning)

As stated above, according to Śabdavyāpāra framework meanings words convey are categorised as vācyārtha (Expressed meaning), lakṣyārtha (Indicated meaning) and vyaṅgyārtha (Suggested meaning). While 'Expressed meaning' is the straightforward sum of meanings of all the words in the sentence, in 'Indicated meaning' or 'Suggested meaning' the meaning of the sentence is not the sum total of the meanings of all the words and differ based on the various nuances of language, local culture etc.,

Expressive: That which denotes the direct conventional (or dictionary) meaning is the Expressive word. In ordinary parlance, a word denotes something by convention of the given language. Where the conventional denotation is not known, there is no comprehension of the meaning. Thus, when the conventional denotation is apprehended directly, without the intervention of any other agency, the word is said to be 'Expressive' of the denotation or meaning. In a sentence the words also need to satisfy three[6] conditions to be able to express the meaningful sentence. They need to have 'mutual requirement' as in all of them are needed, they need to be 'compatible' with each other and there needs to be 'proximity' meaning certain words need to be next to each other. Consider the sentence "*The student is studying mathematics*". It is very clearly understood what each word is denoting, hence each word is expressive. Moreover, they satisfy the three conditions of 'mutual requirement', 'compatibility' and 'proximity'; therefore, the sentence is a meaningful sentence. The meaning of such a sentence obtained by the meanings of the 'Expressive' words is called the 'Expressed'

[5] We picked 300 sentences from a combination of Rabindranath Tagore's Gitanjali, kālidāsa's kumārasambhavam and śāstra texts of dhvanyāloka, kāvyaprakāśa.

[6] आकाङ्क्षा-योग्यता-सन्निधिवशाद् वक्ष्यमाणस्वरूपानां पदार्थानां समन्वये तात्पर्यार्थो विशेषवपुः अपथार्थोपि वाक्यार्थः | – ākāṅkṣā-yogyatā-sannidhivaśād vakṣyamāṇasvarūpānāṃ padārthānāṃ samanvaye tātparyārtho viśeṣavapuḥ apathārthopi vākyārthaḥ When the denotations of different words become related together though 'mutual requirement', compatibility' and 'proximity' there appears in the shape of the 'meaning of the sentence' which is not expressed by any single word constituting the sentence.

meaning. It can also be referred to as the 'Primary' meaning of the sentence.

Indicative [7] : When the 'Primary' (or 'Expressed') meaning does not make sense (because of incompatibility), another meaning, which is in close affinity to what the word is denoting, is implied by the word. Such a meaning is called the 'Indicated' meaning that such a word is the 'Indicative' word in the given sentence. Consider the sentence "*Drunk by the joy of singing, I forget myself*". Here when we put together the 'Primary' meanings of all the words we see there is incompatibility as joy is not a physical drink that one can get drunk on. The word drunk here implies the meaning overtaken or completely filled with. Using this implied meaning of the word drunk, we arrive at the meaning "Overjoyed by singing, I forget myself". This is called the 'Indicated' meaning and the word drunk is the 'Indicative' word in this sentence. This process of implying the 'Indicated' meaning is called 'Indication'. The 'Indicated' meaning of such a sentence makes the import of the sentence much clearer and is also very easily translatable by a machine learning model to any other language

Suggestive: Where the 'Primary' meaning is clear, there can also exist a 'Suggested' meaning. Such a word is called the 'Suggestive' word. The 'Suggested' meaning can also exist along with the 'Indicated' meaning. Since the focus of this paper is on the 'Indicated' meaning and its application, we do not go into the details of this category.

2.1 Various types of 'Indication'

'Usage' and 'Special Purpose' Indication [8] **:**The process of imposing the 'Indicated' meaning is done either based on 'Usage' or for a 'Special Purpose'. and as such these are the 2 categories of 'Indication'.

Example of 'Indication' on the basis of 'Usage': Consider the sentence "*Do not beat around the bush when expressing your viewpoint*". Here the primary meaning of words 'do not beat around the bush' are incompatible with the words 'expressing your viewpoint'. However, it is common usage that means 'do not waste time by giving lengthy and cyclical explanations. This 'Indicated' meaning conveys the meaning of the sentence appropriately. A lot of idioms in English language, for example, fall into this category of 'Indication based on Usage'

Example of 'Indication' on the basis of 'Special Purpose': Consider the sentence "*Her face had blooming smiles at the thought of meeting her lover*". Here the 'Primary' meaning of the word 'blooming' is to be flowering and this is incompatible with the sense of the sentence which is describing the expression of a person's face. The word 'blooming' is implying the 'Indicated' meaning in excess / lot of / big, which is in affinity with its 'Primary' meaning. Read with the 'Indicated' meaning, the sentence means that "Big smiles appeared on her face at the thought of meeting her lover'. An appropriate paraphrased sentence could be 'She had big smiles on her face at the thought of meeting her lover'. Moreover, the implication of the 'Indicated' meaning also has a 'Special Purpose' of referring to the beauty, radiance etc. in an excessive way that appeared on her face at the thought of meeting her lover.

While 'Indication' on the basis of 'Usage' has no further sub-categories, 'Indication' on the basis of 'Special Purpose' has 6 sub-categories.

Six sub-categories of 'Indication' on the basis of 'Special Purpose': 'Indication' on the basis of 'Special Purpose' is further categorised into two[9], namely, 'Pure' and 'Qualitative' Indications. When the 'Indication' relies upon similarity / similitude it is called 'Qualitative' Indication and when it is based upon other kinds of relationships (like cause-effect and not on similarity / similitude) it is called 'Pure' Indication.

'Qualitative' Indication: Consider the sentence "*Her eyes are lotus petals*". In this sentence, the qualities of lotus petals are being imposed upon the eyes of the person and this is to show the similarities in their qualities, for example this lady has big eyes and in the shape of lotus petals. Here the imposed meaning is the quality of the lotus petal that is being imposed upon the eyes of the lady. This is an example of 'Qualitative'

[7] मुख्यार्थबाधे तद्योगे – mukhyārthabādhe tadyoge - When there is incompatibility in the Primary meaning and the other meaning has affinity with the Primary meaning
[8] रूढितोऽथ प्रयोजनात् – rūḍhito'tha prayojanāt - The Indication is of 2 types, based on Usage and based on Special Purpose

[9] भेदाविमौ च सादृश्यात्सम्बन्धान्तरस्तथा bhedāvimau ca sādṛśyātsambandhāntarastathā. These 2 are different. One is by similarity and other by other relationships

Indication. An appropriate paraphrased sentence could be ‘Her eyes are big and beautiful like lotus petals’.

Qualitative Indication can be of two types, namely, ‘Super-imponent Qualitative’ Indication and ‘Intro-susceptive Qualitative’ Indication, based on how the imposed qualities and that which they are being imposed upon are expressed in the sentence.

1. ‘Super-imponent[10] Qualitative’ Indication: When what is being imposed and that which it is being imposed upon are mentioned separately in the sentence it is called a ‘Super-imponent’. The sentence “*Her eyes are lotus petals*” is an example of ‘Super-imponent Qualitative' Indication as what is being imposed (lotus petals) and that which it is being imposed upon (eyes) are mentioned separately. (akin to a simile)

2. ‘Intro-susceptive [11] Qualitative’ Indication: When what is being imposed consumes (takes within itself) that which it is being imposed up on it is called ‘Intro-susceptive’. Both are not mentioned separately in the sentence and only what is being imposed is mentioned. Considering the same example of the lady with big and beautiful eyes, if someone were to look at the lady’s beautiful eyes and say “*They are lotus petals*”, then this becomes an example of ‘Intro-Susceptive Qualitative’ Indication (akin to a metaphor). Here that which is being imposed (lotus petals) has consumed that which it is being imposed upon (eyes). A paraphrased sentence will be “Her eyes, which are big and beautiful, appear to be lotus petals themselves.”

‘Pure’[12] Indication

‘Pure’ Indication is of four types, namely, Inclusive Indication, Indicative Indication, Super-imponent Pure Indication and Intro-susceptive Pure Indication.

3. ‘Inclusive Pure’ Indication: When the implication of the ‘Secondary’ meaning is for the sake of completing the ‘Primary’ meaning itself, it is called Inclusive Indication. Consider the sentence “*Your pizza is on its way*”. Here, the pizza that has been ordered cannot be travelling on its own, there is an unwritten actor present in the sentence, the pizza delivery person. The word pizza without losing its ‘Primary’ meaning is implying an actor to complete the ‘Primary’ meaning itself. This is ‘Inclusive’ Indication. A paraphrased sentence elaborating the Indicated meaning could be “The pizza delivery boy, along with your pizza, is on his way”

4. ‘Indicative Pure’ Indication: When the ‘Primary’ meaning is replaced by the ‘Secondary’ meaning, it is called Indicative Indication. Consider the sentence “*She jumps to conclusions*”. Here, the Primary meaning of the word jumps is replaced by as Secondary meaning ‘to form quickly’. Hence this is an Indicative Indication. A paraphrased sentence elaborating the Indicated meaning will be “She forms conclusions very quickly”

5. Super-imponent Pure Indication: When the Indication is based upon a relationship like cause-effect (and not similarity / similitude) between the imposed and what it is being imposed, and both are stated separately in the sentence it is ‘Super-imponent Pure’ indication. Consider the sentence “*Knowledge is power*”. Here there is a cause-effect relationship between Knowledge (that which it is being imposed upon) and Power (imposed). Moreover, both are being stated clearly in the sentence. Hence it is a ‘Super-imponent Pure’ Indication. An appropriate paraphrased sentence will be “Knowledge gives power”

6. ‘Intro-susceptive Pure’ Indication: This is like the ‘Super-imponent Pure’ Indication but the imposed and that which it is being imposed upon are not stated separately in the sentence. When someone described a knowledgeable person and says “*He has the power*”, it is an example of Intro-susceptive Pure Indication as Power (imposed) consumes the word Knowledge (that which it is being imposed upon) and both the words are stated not stated separately in the sentenc e. An appropriate paraphrased sentence will be “He has the power of knowledge”

[10] सारोपान्या तु यत्रोक्तौ विषयी विषयस्तथा । sāropānyā tu yatroktau viṣayī viṣayastathā . Super-impotent is one where imposed and that which it is being imposed are stated / said separately.

[11] विषयन्तःकृतेन्यस्मिन् सा स्यात्साध्यवसानिका । viṣayantaḥkṛtenyasmin sā syātsādhyavasānikā . That which the imposed consumes that which it is being imposed upon is Intro-susceptive

[12] स्वसिद्धये पराक्षेपः परार्थं स्वसमर्पणम् । उपादानं लक्षणं चेत्युक्ता शुद्धौव सा द्विधा । svasiddhaye parākṣepaḥ parārthaṃ svasamarpaṇam . upādānaṃ lakṣaṇaṃ cetyuktā śuddhauva sā dvidhā . Pure is of 2 types Inclusive and Indicative. Inclusive implies another actor to achieve its Primary meaning. Indicative gives up its Primary meaning to take on the Secondary meaning.

Seven categories of Indication: The two types of ‘Qualitative’ Indication and four[13] types of ‘Pure’ Indication make up the six categories of Indication based on Special Purpose. Along with the ‘Usage based’ Indication there are in total seven categories of Indication. The categorisation helps understanding the sentences and also provides a distinctive way for paraphrasing the sentence for each of the categories. The characteristics of the seven categories discussed above have been summarised into a flow chart presented in Appendix I. A few more examples of the seven categories of Indication are provided in Appendix II.

3 Application of the lakṣyārtha concept to Machine Learning models

If we notice the paraphrasing of the sentences with Indicated meaning, there are patterns that are correlated to the category of the Indicated meaning, in most cases except in the ‘Usage based’ Indication. At a high level the patterns of paraphrasing are summarised in Table 3 below.

Category of Indication	**Typical Pattern of Paraphrase**
Super-imponent Qualitative	Typically ‘like’ or an equivalent word is added in the sentence as the Indication is based on comparison
Intro-susceptive Qualitative	Similar to above along with the addition of the words that are left out in the sentence. This needs context in which the sentence as the speaker would leave out some of the words
Inclusive Pure	A related word(s) are added to explicitly mention the unspoken actor
Indicative Pure	A secondary meaning of the word replaced the word in the sentence where this indication exists. This secondary meaning is typically very closely related to the primary meaning of the word
Super-imponent Pure	Words are added to show the relationship between the imposed and this which it is being imposed. Typically this relationship between the words is quite commonly used
Super-imponent Pure	Same as above along with the addition of the words that are left out in the sentence. This needs context in which the sentence as the speaker would leave out some of the words
Usage based	Does not have any pattern as it is based on widely accepted usage in the given language.

Inspired by the correlation between the paraphrasing and the category of the Indication, we embarked on the pilot of training the Machine Learning models to do this paraphrasing before translation by Google Translate. Given the comprehensiveness and the fundamental nature of the categorisation, we believe that the training can be achieved with relatively small datasets. Hence, we attempted the pilot with a very small dataset. We broke the pilot down into three steps

Table 3: Typical patterns that can be observed in Paraphrasing sentences, based on the category of Indication

Step 1: Identify the existence of an Indicated meaning in the sentence. This means that the model needs to identify the incompatibility between the words in a sentence. To achieve this, we trained a multi-layer perceptron of 3 layers and a 2d Convolutional Neural Network (CNN2d) with filter sizes of 3, 4, 5, 6, 7 and 8 for binary classification on a dataset of 400 example sentences (100 without Indicated meaning and 300 with Indicated meaning). We used 320 of these sentences for training and 80 sentences for testing. While the multi-layer perceptron trained to 70% test accuracy, CNN2d achieved 78% test accuracy. This was on expected lines as the existence of Indicated meaning is identified based on incompatibility between words (refer footnote 4). The CNN2d is comparing groups of adjacent words of length 3, 4, 5, 6, 7 and 8 in its filters to

[13] Commentators of kāvyaprakāśa also explain that Inclusive and Indicative Indications are further divided into Super-impotent and Intro-susceptive each, giving rise to the 4 Pure Indications. Either as per this categorisation or as per what has been explained in Section 2, there are 4 types of Pure Indication. We took the choice of the categorisation that we think is most appropriate for the Machine Translation.

map the incompatibility. Sample classification results by the CNN2d trained on our dataset of 400 sentences are presented below in Table 4.

Sl No	**Sentence**	**Classification – Indication Exists (Yes/No)**	
		Trained CNN2d	**Actual**
1	My heart spreads its wings	Yes	Yes
2	Master is knowledge	Yes	Yes
3	He is speaking the truth	Yes	No
4	Truth is bitter to swallow	No	Yes
5	I can run fast	No	No
6	On this stormy night the sky groans	Yes	Yes
7	It is raining heavily today	No	No
8	He is a walking encyclopeadia	Yes	Yes
9	He stole her heart	Yes	Yes
10	The water is blue in colour	No	No

Table 4: CNN2d classification of whether an Indicated meaning exists in the given sentence

Step 2: Identify the category of Indicated meaning in a sentence. We labelled the 400 sentences with eight labels (one label for Expressive and one each for category of Indication per the framework explained in Section 2) and trained the CNN2d for multi-classification. 320 sentences were used for training and 80 were used for testing. The model achieved 72% test accuracy. Sample classification results by the CNN2d trained on our dataset are presented below in Table 5.

Sl No	**Sentence**	**Category of Indication**	
		Trained CNN2d	**Actual**
1	He is stretching the truth	Indicative Pure	Indicative Pure
2	He is a walking encyclopeadia	Usage Based	Super-imponent Qualitative
3	On this stormy night the sky groans	Inclusive Pure	Inclusive Pure
4	This is a magnificent new shirt	Expressive	Expressive
5	Health is wealth	Super-imponent Qualitative	Super-imponent Qualitative
6	His radiance was visible from far	Indicative Pure	Indicative Pure
7	The bus is arriving late	Inclusive Pure	Inclusive Pure
8	Master is knowledge	Super-imponent Qualitative	Super-imponent Pure
9	My heart spreads its wings	Super-imponent Qualitative	Intro-susceptive Qualitative
10	Truth is bitter to swallow	Expressive	Indicative Pure
11	Your pizza is on its way	Expressive	Inclusive Pure
12	Time heals everyone	Intro-susceptive Qualitative	Intro-susceptive Qualitative

Table 5: CNN2d classification of a given sentence based on the type of Indicated meaning it contains

Step 3: Paraphrase the sentence with elaborating the Indicated meaning based on the category of Indicated meaning. We finetuned a pre-trained Google's T5 model to paraphrase sentences with our custom dataset of sentences with Indicated meaning. We used the patterns described in Table 3 to create our custom dataset. Our dataset contained 250 training sentences and 50 testing sentences. We refer to this finetuned T5 model as the T5-I.

We then used the T5-I paraphrased sentences as input to Google Translate for translation to various languages. We then used BLEU [14] (Bilingual Evaluation Understudy) score to compare the translation with and without T5-I paraphrasing. For the purpose of this pilot we used translation to three Indian languages – Telugu, Hindi, Kannada. We used a typical human translation of the sentences in these 3 languages as reference for calculation the BLEU scores. Here, we present the comparison of translation with and without the paraphrasing by T5-I for a few validation sentences along with the respective BLEU scores.

Original Sentence: She was showered with blessings			
Expected Translation: ఆమె చాలా ఆశీస్సులు పొందింది. / उसको बहुत सारे आशीर्वाद् मिले/ ಅವಳಿಗೆ ಬಹಳಷ್ಟು ಆಶೀರ್ವಾದಗಳು ದೊರಕಿದವು			
Paraphrased Sentence (by T5-I) : She received lots of blessings			
Google Translate's Translation Of			
Original Sentence	**BLEU Score**	**Paraphrased Sentence**	**BLEU Score**
ఆమె ఆశీస్సులతో ముంచెత్తింది	0.54	ఆమె చాలా దీవెనలు పొందింది	0.70
वह आशीर्वाद के साथ नहाया गया था	0	उन्हें बहुत आशीर्वाद मिला	0.55
ಅವಳು ಆಶೀರ್ವಾದದಿಂದ ಸುರಿಸಲ್ಪಟ್ಟಳು	0	ಅವಳು ಬಹಳಷ್ಟು ಆಶೀರ್ವಾದಗಳನ್ನು ಪಡೆದಳು	0.70

Original Sentence: He stretched the truth			
Expected Translation: అతడు అబద్ధమ ఆడాడు / उसने झूठ बोला / ಅವನು ಸುಳ್ಳನ್ನು ಹೇಳಿದನು			
Paraphrased Sentence (by T5-I) : He used falsehood			
Google Translate's Translation Of			
Original Sentence	**BLEU Score**	**Paraphrased Sentence**	**BLEU Score**
సత్యాన్ని సాగదీశాడు	0	అతను అసత్యాన్ని ఉపయోగించాడు	0.76
उसने सच फैलाया	0.76	उन्होंने झूठ का इस्तेमाल किया	0.66
ಅವರು ಸತ್ಯವನ್ನು ವಿಸ್ತರಿಸಿದರು	0	ಅವರು ಸುಳ್ಳನ್ನು ಬಳಸಿದರು	0.76

Original Sentence: I buy peace of mind by being silent			
Expected Translation: నేను మోనంగా ఉండి మనశ్శాంతిని పొందుతాను / मुझे चुप रहकर सुकून मिलता है / ನಾನು ಮೌನದಿಂದ ನೆಮ್ಮದಿಯನ್ನು ಗಳಿಸಿದೆ.			
Paraphrased Sentence (by T5-I) : My peace of mind comes by being silent			
Google Translate's Translation Of			
Original Sentence	**BLEU Score**	**Paraphrased Sentence**	**BLEU Score**
నేను మౌనంగా ఉండడం ద్వారా మనశ్శాంతిని కొనుక్కుంటాను	0.79	మౌనంగా ఉండడం వల్ల నా మనశ్శాంతి కలుగుతుంది	0.0
मैं चुप रहकर मन की शांति खरीदता हूँ	0.43	मेरे चुप रहने से आती है मन की शांति	0.68
ನಾನು ಮೌನವಾಗಿರುವುದರ ಮೂಲಕ	0.64	ಮೌನದಿಂದ ನನ್ನ ಮನಸ್ಸಿಗೆ	0.69

[14] BLEU (BiLingual Evaluation Understudy) is a metric for automatically evaluating machine-translated text. The BLEU score is a number between zero and one that measures the similarity of the machine-translated text to a set of high quality reference translations. A value of 0 means that the machine-translated output has no overlap with the reference translation (low quality) while a value of 1 means there is perfect overlap with the reference translations (high quality).

ಮನಸ್ಸಿನ ಶಾಂತಿಯನ್ನು ಖರೀದಿಸುತ್ತೇನೆ		ಶಾಂತಿ ಸಿಗುತ್ತದೆ	

Original Sentence: An idea sprouted in his mind			
Expected Translation: అతనికి ఒక ఆలోచన వచ్చింది / उसको मन में एक विचार आया / ಅವನಲ್ಲಿ ಒಂದು ಉಪಾಯವು ಚಿಗುರೊಡೆಯಿತು.			
Paraphrased Sentence (by T5-I) : An idea came to his mind			
Google Translate's Translation Of			
Original Sentence	**BLEU Score**	**Paraphrased Sentence**	**BLEU Score**
అతని మదిలో ఒక ఆలోచన మొలకెత్తింది	0.56	అతని మదిలో ఒక ఆలోచన వచ్చింది	0.56
उसके मन में एक विचार कौंधा	0.50	उसके दिमाग में एक विचार आया	0.48
ಅವನ ಮನಸ್ಸಿನಲ್ಲಿ ಒಂದು ಕಲ್ಪನೆ ಚಿಗುರೊಡೆಯಿತು	0.79	ಅವನ ಮನಸ್ಸಿಗೆ ಒಂದು ಉಪಾಯ ಹೊಳೆಯಿತು	0.67

It can be noticed that even where the individual BLEU score did not show improvement with the paraphrasing by T5-I, the translation of the paraphrased sentence is much more meaningful than the one without. Taking a corpus score on the 12 Google Translate translations above the BLEU score for translations improved from 0.39 to 0.6 with paraphrasing by T5-I.

4 Conclusion

Where the CNN2d correctly identified the existence of Indication, it performed very well in identifying the sub-categories of Indication except in the case of the two Super-imponent Indications. We believe this because of the lack of equal number of examples across categories in our training dataset. The improvement in BLEU score achieved for translations with paraphrasing by T5-I is significant and encouraging. The difference in the translation with and without T5-I paraphrasing was very evident in more complicated literary usages of sentences (and not just metaphors).

For the purpose of the pilot we trained black-box[15] implementations of CNN2d and T5-base in with a small dataset. We believe fine-tuning of the model architecture and a limited increase in the dataset can improve accuracy of the models for paraphrasing and translating sentences in Literary works with Indicated meaning to a higher level of accuracy. Where word embeddings are available the trained models should also work across languages..

There are other use cases as well, where understanding the real intent of a sentence depends on understanding of the Indicated meaning, including dialogue systems, sentiment analysis and emotion analysis.

5 Acknowledgements

The authors would like to acknowledge the efforts of Dr Sai Susarla in reviewing the methodology and providing critical inputs, Sriram Subramanian in helping with creation of dataset and providing review comments for the paper, and Manjula Pilaka for providing guidance in the programming that was needed to train and test the machine learning models.

6 References

Almahasees, Zakaryia Mustafa, Machine Translation Quality of Khalil Gibran's the Prophet (November 9, 2017). AWEJ for translation & Literary Studies Volume, 1 Number 4, October 2017

[15] We leveraged existing code of CNN2d and T5 models from Google Colab's (Colaboratory is cloud based ML resource) pytorch libraries.

Anandavardhanacharya, Dhvanyāloka, Editor, Acharya Jagannath Pathak, Chowkambha Vidyabhavan, Varanasi, 2009

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In Proceedings of the International Conference on Learning Representations (ICLR), May.

Besacier, Laurent and Lane Schwartz. 2015. Automated translation of a literary work: a pilot study. In Proceedings of the Fourth Workshop on Computational Linguistics for Literature, pages 114–122.

Acharya Mammata, Kāvyaprakāśa, , Editor Acharya Satyanarayanasastri Khanduri, Chawkambha Krishnadas Academy, Varanasi, 2016

James Hadley, Maja Popović, Haithem Afli, Andy Way (Editors), Proceedings of Qualities of Literary Machine Translation, Machine Translation Summit XVII, August 2019, Dublin, Ireland

Jerry Lui, N. O. (2020). Metaphor Detection Using Contextual Word Embeddings From Transformers. Proceedings of the Second Workshop of Figurative Language Processing. Association of Computational Linguistics

M. R. Kale, kumārasambhavam of kālidāsa – English Translation, Second Edition,The Standard Publishing Co. Bombay, 1917

Mahamahopadhyaya Dr. Sir Ganganatha Jha kāvyaprakāśa, of Mammata with English Translation, Chapters I to X, Bharatiya Vidya Prakasan, 1966

Matusov, E. (2019). The Challenges Of Using Neural Machine Translation for Literature. The Qualities of Literary Machine Translation. Dublin.

P. V. Kane, History of Sanskrit Poetics, Motilal Banarasidass Publishers Pvt. Ltd. Delhi, 2005

Rabinovich, Ella, Shachar Mirkin, Raj Nath Patel, Lucia Specia, and Shuly Wintner. 2016. Personalized machine translation: Preserving original author traits. arXiv preprint arXiv:1610.05461.

Sennrich, Rico, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. pages 86–96, August

Rob Voigt, D. J. (2012). Towards a Literary Machine Translation: The Role of Referential Cohesion. Workshop on Computational Linguistics for Literature (pp. 18-25). Montreal, Canada: Association for Computational Linguistics.

Rui Mao, C. L. (2018). Word Embedding and WordNet Based Metaphor Identification and Interpretation. Proceedings of 56th Annual Meeting of the Association for Computational Linguistics. Melbourne: Association for Computational Linguistics.

Sir Rabindranath Tagore, Gitanjali, English Translation, 1912

Taja Kuzman, Špela Vintar, Mihael Arčan, Machine, Neural Machine Translation of Literary Texts from English to Slovene-Translation Summit XVII, August 2019, Dublin, Ireland

Vamanacarya Ramabhatta Jhalakikara , bālabodhini Samskrit commentary on kāvyaprakāśa, Second Edition, Government Central Book Depot 1901

EduMT: Developing Machine Translation System for Educational Content in Indian Languages

Ramakrishna Appicharla, Asif Ekbal, Pushpak Bhattacharyya
Department of Computer Science and Engineering
Indian Institute of Technology Patna
Patna, Bihar, India
{appicharla_2021cs01,asif,pb}@iitp.ac.in

Abstract

In this paper, we explore various approaches to build Hindi to Bengali Neural Machine Translation (NMT) systems for the educational domain. Translation of educational content poses several challenges, such as unavailability of gold standard data for model building, extensive uses of domain-specific terms, as well as the presence of noise in the form of spontaneous speech as the corpus is prepared from subtitle data and noise due to the process of corpus creation through back-translation. We create an educational parallel corpus by crawling lecture subtitles and translating them into Hindi and Bengali using Google translate. We also create a clean parallel corpus by post-editing synthetic corpus via annotation and crowd-sourcing. We build NMT systems on the prepared corpus with domain adaptation objectives. We also explore data augmentation methods by automatically cleaning synthetic corpus and using it to further train the models. We experiment with combining domain adaptation objective with multilingual NMT. We report BLEU and TER scores of all the models on a manually created Hindi-Bengali educational testset. Our experiments show that the multilingual domain adaptation model outperforms all the other models by achieving 34.8 BLEU and 0.466 TER scores.

1 Introduction

Massive Open Online Courses (MOOCs) have gained a lot of attention in recent years due to the availability of high-quality educational resources free of cost. In India, National Programme on Technology Enhanced Learning (NPTEL)[1] is one such initiative to promote online education. However, most of the content offered in English poses a problem for non-native English language speakers especially in a multilingual country like India. One potential solution to mitigate this problem is developing Machine Translation (MT) systems to translate contents from English to other Indian languages. Developing Machine Translation (MT) systems between two Indian languages is more difficult than developing systems between English and Indian languages due to the unavailability of the educational parallel corpus for Indian languages. MT systems, especially current state-of-the-art Neural Machine Translation (NMT) systems (Sutskever et al., 2014; Bahdanau et al., 2015; Vaswani et al., 2017) are data-hungry and requires a lot of training data (Zoph et al., 2016; Koehn and Knowles, 2017). Developing MT systems for the educational domain poses issues such as lack of data, translation of domain-specific terms, phrases, and mathematical expressions. Since the dataset is prepared from the lecture subtitles and transcripts, it also contains noise in the form of spontaneous speech (e.g: 'umm', 'yes! good morning' etc.) and repetition of phrases (e.g: 'ok, well.. ok well, now we have to compute this value' etc.). Due to these issues, building an MT system for the educational domain is a challenging task.

In this paper, we focus on developing the NMT systems between two Indian languages, namely Hindi $\rightarrow$ Bengali language pair for the computer science domain. We create a Hindi-Bengali educational corpus by crawling NPTEL lecture subtitles, transcripts that are in English and translating them into Hindi and Bengali. We create two types of educational parallel corpora, *'synthetic'* and *'clean'*. *Synthetic* corpus is prepared from translating English data into Hindi and Bengali with the help of Google Translate[2]. *Clean* corpus is prepared by manual post-editing of synthetic data via manual annotation and crowd-sourcing. We conduct experiments on prepared corpora with domain adapta-

[1]https://nptel.ac.in/

[2]https://translate.google.com/

tion (Chu et al., 2017), objective. We experiment with denoising (Edunov et al., 2018) and automatic post-editing (Pal et al., 2016) objectives to automatically clean the *synthetic* corpus which are further used to train the models. We also experimented on multilingual NMT (Johnson et al., 2017) using the English part of the corpus along with Hindi and Bengali. Since there is no standard educational test set is available to test our models' performance, we manually create the test set by translating the Hindi part of the English-Hindi parallel corpus (AI domain) from Adap-MT shared task (Sharma et al., 2020) into Bengali. We report BLEU and TER score (Post, 2018) on the prepared test corpus[3].

The paper is organized as follows. In Section 2, we briefly review a few of the notable works on building MT systems for educational content and domain adaptation, data augmentation methods in NMT. In Section 3, we describe the corpus creation process. The experimental setup used to conduct domain adaptation and semi-synthetic data augmentation experiments are described in Section 4 and Section 5, respectively. The NMT model settings and experimental setup are described in Section 6. Results are described in Section 7. Finally, the work is concluded in Section 8.

2 Related Work

Building an NMT system for any domain requires a significant amount of data. In the educational domain, obtaining data is very challenging. Most of the works in building MT systems for the educational domain is focused on creating corpora. Abdelali et al. (2014) have created educational corpora for 20 languages (20 monolingual and 190 parallel corpora) by crawling AMARA website[4] which is a community-driven web-based platform for editing and creating subtitles for videos. Parallel corpus for European languages in Educational domain have been created via crowd-sourcing Kordoni et al. (2016); Sosoni et al. (2018); Behnke et al. (2018) . They built NMT systems on the prepared corpora and report that even a small amount of crowd-sourced translations can improve the translation quality.

Domain adaptation is a methodology to adapt models trained on out-of-domain data to in-domain data. Chu et al. (2017) proposed two methods for fine-tuning which do not need any modifications to standard NMT architecture. One method is to add domain tags (e.g: '<2domain>') and train the NMT model on the combined corpora from multiple domains. The second method is to fine-tune the model trained on out-of-domain data on the combination of in-domain and out-of-domain data. Britz et al. (2017) proposed three methods for domain adaptation. *'Discriminative Mixing'* method uses a discriminator which is a fully connected layer, to predict the domain tag the current input sentence belongs to. The loss from discriminator and decoder is added and back-propagated which jointly optimizes the network. This makes the encoder encode domain-related features. *'Adversarial Discriminative Mixing'* method is the same as *'Discriminative Mixing'* method except while back-propagating loss, the loss from discriminator is reversed by multiplying it with -1. This makes the encoder encode domain invariant features. *'Target Token Mixing'* does not use a discriminator network but simulates the discriminator by adding domain tags to the target sentence.

Improving the performance of NMT models with additional monolingual data is a common practice especially in low-resource settings. Back-translation (Sennrich et al., 2016) is an effective approach to make use of the target monolingual data. Edunov et al. (2018) conducted various experiments to generate synthetic source sentences from target monolingual data and used it to further train the models. They report that corrupting synthetic source sentences with noise and using that noisy source sentence instead of a clean synthetic source sentence, significantly improve the performance of the NMT models. Multilingual NMT (Johnson et al., 2017) is another popular approach to improve the performance of NMT models for low resource language pairs by augmenting the low resource pairs with high resource language pairs and training a single NMT model.

In this work, we build NMT models with domain adaptation objectives. We experiment with cleaning synthetic in-domain corpus with denoising auto-encoder (Vincent et al., 2008) and Automatic Post-Editing (Pal et al., 2016) objectives. The resulting data is augmented with created in-domain corpus and used to train NMT models. We also experiment with combining multilingual NMT and domain adaptation objectives.

[3]The developed system can be accessible via following link: http://edumt.ngrok.io/

[4]https://amara.org/en/

3 Corpus Creation

We prepare the parallel corpus in educational domain by crawling lecture subtitles. Specifically, we crawl the lecture subtitles from YouTube and lecture transcripts from NPTEL courses[5]. The subtitles crawled from YouTube are of smaller length compared to subtitles crawled from lecture transcripts. We crawl lectures on Programming, Data Structures, Algorithms, Machine Learning, and Artificial Intelligence.

3.1 Data Crawling

- **Crawling Subtitles:** Video lecture subtitles are crawled from NPTEL[6] and MIT OCW[7] YouTube channels. We crawl the data using *youtube-transcript-api*[8] Python package. First, we collect the URLs of lecture videos. Every video has two types of subtitles in English. One is auto-generated by YouTube and the second one is official subtitles uploaded along with the video. We extract only the official subtitles to minimize the amount of noise in the data as much as possible. Table 1 shows the statistics of crawled subtitle corpus.

- **Crawling Lecture Transcripts:** Lecture transcripts are crawled from the NPTEL courses. For a given course, the lecture transcript is made available in PDF format. Every PDF is tagged as 'Verified' and 'To be verified'. We consider the courses whose transcripts are tagged as 'Verified'[9]. We use *pdftotext*[10] Python package to extract text from PDF. After getting the text, we use *sacremoses*[11], a Python implementation of Moses (Koehn et al., 2007) tokenizer to tokenize the data into sentences. Table 2 shows the statistics of crawled transcript corpus.

3.2 Creation of Synthetic Corpus

The crawled data is in English. To prepare the Hindi-Bengali parallel corpus, we use Google translate tool. The lecture subtitles are crawled from YouTube, translated into Hindi and Bengali with the help of YouTube's built-in Google translate tool. The lecture transcripts are translated into Hindi and Bengali with the help of Google translate web interface[12]. Table 3 shows the statistics of prepared synthetic corpus. Table 4 shows language-wise average sentence length of synthetic corpus prepared from the subtitles and transcripts.

Domain	**#Videos**	**#Subtitles**
Prog, DS and Algo	838	263,150
ML and AI	678	176,764
Total	1,516	439,914

Table 1: Statistics of corpus prepared from YouTube subtitles. Here, Prog: Programming, DS: Data Structures, Algo: Algorithms, ML: Machine Learning, AI: Artificial Intelligence. #Videos: No. of videos and #Subtitles: No. of subtitles.

Domain	**#PDFs**	**#Subtitles**
Prog, DS and Algo	324	46,009
ML and AI	775	109,179
Total	1,099	155,188

Table 2: Statistics of corpus prepared from NPTEL lecture transcripts. Prog: Programming, DS: Data Structures, Algo: Algorithms, ML: Machine Learning, AI: Artificial Intelligence. #PDFs: No. of transcript PDFs and #Subtitles: No. of subtitles.

Domain	**#Subtitles**
Prog, DS and Algo	309,159
ML and AI	285,943
Total	595,102

Table 3: Statistics of the synthetic corpus. Prog: Programming, DS: Data Structures, Algo: Algorithms, ML: Machine Learning, AI: Artificial Intelligence. #Subtitles: No. of subtitles.

Language	**Subtitles**	**Transcripts**
Bengali	11.7	13.96
Hindi	14.6	17.57
English	13.61	16.24

Table 4: Average sentence lengths of synthetic corpora for each language. Subtitles: Data crawled from YouTube lecture subtitles. Transcripts: Data crawled from NPTEL lecture transcripts.

[5]https://nptel.ac.in/course.html

[6]https://www.youtube.com/user/nptelhrd

[7]https://www.youtube.com/user/MIT

[8]https://pypi.org/project/youtube-transcript-api/

[9]'Verified' transcripts are the transcripts that are post-edited after the automatic transcription is done.

[10]https://github.com/jalan/pdftotext

[11]https://github.com/alvations/sacremoses

[12]translation using Google translate is done between July 2020 to February 2021.

3.3 Creation of Clean Corpus

We create a clean Hindi-Bengali parallel corpus by taking part of synthetic corpus and post-edited by annotators and crowd-sourcing. We remove this data from the synthetic corpus to avoid data duplication when training models. We employ three annotators who are fluent in English, Hindi, and Bengali. We provide English corpus and corresponding Hindi and Bengali translations. The annotators post-edited both Hindi and Bengali data based on the English data. We follow the same method to get data post-edited by crowd-sourcing[13] company also. After a clean corpus is created, we took a random sample of 263 Hindi-Bengali parallel sentences for analysis. We ask 4 people who speak both Hindi and Bengali[14] to score the random sample based on Adequacy and Fluency on a scale of 1-5. For the Hindi part of the sample, the average adequacy and fluency scores are 4.3 and 4.5, respectively. For the Bengali part of the sample, the average adequacy and fluency scores are 4.3 and 4.6, respectively. Based on the manual analysis of the post-edited corpus, we conclude that the post-edited clean corpus is of high quality. Table 5 shows the statistics of the clean corpus.

Domain	#Subtitles
Prog, DS and Algo	22,046
ML and AI	18,190
Total	40,236

Table 5: Statistics of the clean corpus. Here, Prog: Programming, DS: Data Structures, Algo: Algorithms, ML: Machine Learning, AI: Artificial Intelligence. #Subtitles: No. of subtitles.

Corpus	Domain	#Sentences
Synthetic + Clean	Educational	635,338
Samanantar	General	2,501,608

Table 6: Statistics of data used in experiments. Here, Synthetic: Prepared synthetic educational corpus. Clean: Prepared clean educational corpus. Samanantar: Samanantar Hindi-Bengali corpus. #Sentences: No. of sentences.

4 Domain Adaptation

We consider both synthetic and clean Hindi-Bengali educational parallel corpus as in-domain data. Samanantar corpus (Ramesh et al., 2021)[15] is considered as out-of-domain data. Table 6 shows the statistics of data used in experiments. Since there is no standard Hindi-Bengali educational test set is available to test our models, we manually create the test set by translating Hindi part of English-Hindi parallel corpus (AI domain) from Adap-MT shared task (Sharma et al., 2020) into Bengali. We carefully create the test set by avoiding any overlap between the test set and in-domain corpus which is used for training. The prepared test set of size 2,630 sentences is used to evaluate all trained models.

We train two baseline models, namely 'Out-of-domain baseline' and 'In-domain baseline'. The out-of-domain baseline model is trained on Samanantar corpus and the in-domain baseline model is trained on the prepared clean educational parallel corpus. We train two domain adaptation models by following fine-tuning (Chu et al., 2017) method. Specifically, we use the out-of-domain baseline model as the parent model. The parent model is fine-tuned with (i). Clean educational parallel corpus (denoted as 'FT-Clean') (ii). Synthetic + Clean educational parallel corpus (denoted as 'FT-Both'). The reason to build two fine-tuned models is to check whether synthetic corpus is improving model performance or not. Based on the results (ref Table 7) we choose to use both Synthetic and Clean parallel corpus as our in-domain corpus. We also train another fine-tuned model following mixed fine-tuning (Chu et al., 2017) method. Similar to fine-tuned models, the out-of-domain baseline is used as a parent model and fine-tuned with the combination of Samanantar and Synthetic + Clean educational parallel corpus (denoted as 'FT-Both-Mixed').

We also experiment with adding domain tags[16] to source sentence (Chu et al., 2017) (denoted as 'Source Token Mixing') and target sentence (Britz et al., 2017) (denoted as 'Target Token Mixing'). Using these methods, a single model can be trained on both out-of-domain and in-domain data at the same time. This will save time to train the model. In our case, since out-of-domain data size is very large compared to in-domain data, we oversample in-domain data to match the size of the out-of-domain data.

[13]https://xsaras.com/

[14]Please note that there is no overlap between annotators who post-edited the corpus and evaluators.

[15]https://indicnlp.ai4bharat.org/samanantar/#indic-indic

[16]We use ##2GEN, ##2EDU tags to denote general and educational domains respectively.

4.1 Multilingual Domain Adaptation

Multilingual NMT model (Johnson et al., 2017; Sen et al., 2018) is a single model trained for multiple translation directions by combining parallel corpora from multiple languages into a single unified corpus. Multilingual models have shown improvement for language pairs having less corpus. In this work, we experiment with combining domain adaptation objective with multilingual model (Chu and Dabre, 2019) to check whether adding another language to the corpus will improve the model performance or not (denoted as 'FT-Multilingual'). To build this model, we use the Out-of-domain baseline model which is trained on Hindi-Bengali Samanantar corpus, as the parent model. We fine-tune the model on multilingual in-domain corpus obtained by combining Hindi-Bengali, Hindi-English, and English-Bengali corpus[17]. Specifically, we concatenated Hindi-English, English-Bengali, and Hindi-Bengali corpora. Similar to Johnson et al. (2017), we use language tags to denote the target language[18]. Here, the English part of the corpus act as a bridge between Hindi and Bengali.

5 Semi-Synthetic Data Augmentation

Since most of our in-domain data is synthetic, we conduct experiments on automatic corpus cleaning. We experiment with two methods for automatic corpus cleaning, Denoising auto-encoder (Vincent et al., 2008; Edunov et al., 2018) and Automatic Post-Editing (APE) (Pal et al., 2016). We conduct experiments on the Bengali part of the corpus as it is our target language. We use synthetic-clean Bengali sentence pairs from Clean corpus[19] as our training corpus for corpus cleaning experiments. With the APE objective, we train an end-to-end NMT model with synthetic Bengali sentences as input and clean Bengali sentences as the target. Edunov et al. (2018) show that when using back-translated (Sennrich et al., 2016) data to train the NMT model, adding noise to input sentences improve model performance significantly. Similarly, we create a noisy version of source sentences with two types of noise: (i). Randomly dropping word with probability 0.1 (ii). randomly swapping tokens with its neighboring token with probability 0.1 (Edunov et al., 2018). We do not modify the target sentences. We also experiment by combining these two objectives and training a single model which can perform both denoising and automatic post-editing. After training, we use these models to generate clean Bengali sentences from synthetic Bengali sentences. We denote this as 'Semi-Synthetic' corpus since the source (Hindi) is synthetic and the target (Bengali) is automatically cleaned.

The main reason to perform automatic corpus cleaning is to use the resulting clean corpus to improve the performance of the NMT model for the educational domain. To this extent, we repeat the experiment similar to 'FT-Both' which is fine-tuning the model trained on Samanantar corpus with educational corpus. However, now we use Semi-Synthetic corpus along with Synthetic and Clean corpora to fine-tune the model. 'FT-Both + Denoising' denotes the model fine-tuned with clean, synthetic corpora and semi-synthetic corpus obtained from the denoising experiment. 'FT-Both + APE' denotes the model fine-tuned with clean, synthetic corpora and semi-synthetic corpus obtained from the APE experiment. Similarly, 'FT-Both + Denoising + APE' denotes the model fine-tuned with clean, synthetic corpora and semi-synthetic corpus obtained from the experiment combining denoising and APE objectives. The reason to combine the semi-synthetic data with clean and synthetic data is to provide the model with as much data as possible since in-domain data size is less compared to out-of-domain data.

6 Experimental Setup

All the models have trained on the Transformer (Vaswani et al., 2017) architecture. We use 6 layer Encoder-Decoder stacks with 8 attention heads. Embedding and hidden sizes are set to 512, dropout (Srivastava et al., 2014) rate is set to 0.1. The feed-forward layer consists of 2,048 cells. Adam (Kingma and Ba, 2015) optimizer is used for training with 8,000 warm-up steps with an initial learning rate of 2. We use token-wise batching with batch size set to 2048 tokens. For fine-tuned models, the parent model is trained till convergence[20] and the child model is initialized with the last checkpoint from the parent model without resetting any hyper-parameters. All the models are trained

[17]Since we created the educational corpus by translating English to Hindi and Bengali, we have 3-way parallel corpus involving Hindi, Bengali and English languages

[18]We use ##2EN, ##2BN tags to denote English and Bengali respectively

[19]Since we created clean corpus from synthetic corpus, we have synthetic-clean sentence pairs.

[20]Perplexity is used as stopping criterion.

till convergence and checkpoints are created after every 10,000 steps. All the checkpoints are averaged and considered the best parameters for the respective model. We use OpenNMT toolkit (Klein et al., 2017)[21] to train the models. We tokenize the data into subwords with the unigram language model (Kudo, 2018) using SentencePiece (Kudo and Richardson, 2018) implementation. For all the models except 'FT-Multilingual', we learn subword rules on corpus obtained by concatenating in-domain and out-of-domain corpora. The size of subword vocabulary is 50K for both Hindi and Bengali. For the 'FT-Multilingual' model, we learn joint subword vocabulary for Hindi, Bengali, and English by combining all the in-domain corpora and Hindi-Bengali out-of-domain corpora, and the size of joint subword vocabulary is 75K. At the time of decoding, the beam size is set to 5 with no length penalty.

Model	**BLEU(↑)**	**TER(↓)**
Out-of-domain Baseline	17.3	0.608
In-domain Baseline	12.6	0.704
FT-Clean	21.5	0.634
FT-Both	33.6	0.482
FT-Both-Mixed	27.7	0.548
Source Token Mixing	23.0	0.607
Target Token Mixing	18.6	0.692
FT-Multilingual	**34.8**	**0.466**
FT-Both + Denoising	33.5	0.481
FT-Both + APE	33.0	0.493
FT-Both + Denoising + APE	32.7	0.493

Table 7: BLEU and TER scores of all trained models. FT-Multilingual model outperforms all other models with 34.8 BLEU score and 0.466 TER score.

7 Results and Analysis

We test all the models on the prepared Hindi-Bengali test corpus of size 2,630 and report BLEU (Papineni et al., 2002) and TER (Snover et al., 2006) scores, calculated with sacreBLEU (Post, 2018)[22]. Table 7 shows the results of the models[23]. The two baseline models, *viz.* Out-of-domain Baseline and In-domain Baseline performance are the lowest of all the other models. This behavior is expected since there is less relevant data as the models are trained on out-of-domain corpus and small in-domain corpus respectively. However, the models trained with fine-tuning objectives, namely FT-Clean, FT-Both, and FT-Both-Mixed achieve better results than both baseline models. Specifically, FT-Both, the model which fine-tuned with both clean and synthetic in-domain corpus achieved better results than the other two models. Interestingly, FT-Both-Mixed, the model fine-tuned with combining data from in-domain and out-of-domain data, achieves less BLEU score (27.7) than FT-Both (33.6) despite this method showing improvement (Chu et al., 2017) in other cases. In our case, adding the out-of-domain data is not helping the model but when compared to the other two fine-tuned models, it converged faster which suggests that the model is over-fitting. We also observe that adding in-domain data although it is synthetic, is helping the model.

The models, Source Token Mixing and Target Token Mixing performance are less compared to fine-tuned models. Despite a single model jointly trained for both in-domain and out-of-domain and can share information between both the domains, performance on in-domain data is not significant. Both the models outperform baseline models but the Target Token Mixing model achieves less BLEU score (18.6) than the FT-Clean model (21.5). Similar to the FT-Both-Mixed model, adding out-of-domain data is acting as noise which limits the performance of the model on in-domain data.

The model fine-tuned with the multilingual educational corpus (FT-Multilingual) achieve the highest BLEU score of 34.8 and lowest TER score of 0.466 (higher BLEU and lower TER scores are preferable) of all other models. We observe that adding more in-domain data is improving the model performance. In our case, we add English in-domain corpus (i.e. Hindi-English and English-Bengali) to the Hindi-Bengali corpus. Since both Hindi and Bengali synthetic data were prepared from English data, adding English along with Hindi-Bengali helped the model to learn better representations for the Hindi-Bengali pair. This is evident from the experiments with BLEU score of FT-Multilingual model (34.8) improved by +1.2 points than FT-Both model (33.6). Similarly the TER score of the FT-Multilingual model (0.466) improved by -0.016[24] points than FT-Both model (0.482).

[21]https://github.com/OpenNMT/OpenNMT-py/tree/1.2.0

[22]https://github.com/mjpost/sacreBLEU

[23]sacreBLEU signatures:
BLEU+case.mixed+numrefs.1+smooth.exp+tok.13a+version.1.5.1
TER+tok.tercom-nonorm-punct-noasian-uncased+version.1.5.1

[24]Negative sign indicates the improvement as lower TER score is better.

Results from the semi-synthetic in-domain data augmentation models are interesting due to the reason that adding more in-domain data is not improving the performance. This observation is opposite of the observation from the FT-Multilingual model where adding the English part of the parallel corpus is making the model outperform all other models. Although the models, namely FT-Both + Denoising, FT-Both + APE, and FT-Both + Denoising + APE are trained on in-domain corpus twice the size of actual in-domain corpus (since we add the semi-synthetic corpus to clean and synthetic corpus, the size of in-domain corpus become almost doubled) none of the models can outperform FT-Both model (it only trained on clean and synthetic corpus). However, these three models outperform all other models except FT-Both and FT-Multilingual with FT-Both + Denoising model achieving the second-best TER score (0.481). We observe that the Denoising objective is more effective than the APE objective for automatic corpus cleaning. We believe that if more synthetic-clean in-domain sentence pairs are available to train the denoising model, it will improve the quality of the semi-synthetic corpus which, in turn, improves the NMT model.

We conduct a human evaluation on the output of our best model, namely FT-Multilingual. We randomly choose 50 sentences from the test set and given to 4 evaluators[25] along with reference and output of the model and asked to evaluate based on Adequacy and Fluency on the scale of 1-5. The average adequacy and fluency scores are 3.5 and 3.85, respectively. Based on the human evaluation, we conclude that the model can translate educational data with good adequacy and fluency.

8 Conclusion

In this paper, we have explored the problem of building an NMT system in the educational domain for the Hindi-Bengali language pair. Since there is no data available in the educational domain, we created the parallel corpus by extracting from lecture subtitles and transcripts and translating them into Hindi and Bengali. We also create a clean parallel corpus by post-editing the parallel corpus via crowd-sourcing as well as with the help of annotators. We trained Neural Machine Translation models with domain adaptation objectives by training models on publicly available Samanantar Hindi-Bengali parallel corpus and fine-tuned with prepared educational data. We explored various methods to fine-tune the models such as mixed fine-tuning, source token mixing, and target token mixing. We experimented with data augmentation methods by automatically cleaning the synthetic in-domain corpus with denoising auto-encoder and automatic post-editing objectives. The resulting data is combined with prepared in-domain corpus and trained models. We also experimented with combining domain adaptation with multilingual NMT by training a model on Samanantar Hindi-Bengali corpus and fine-tuned with multilingual in-domain corpus obtained by combining Hindi-Bengali, Hindi-English, and English-Bengali in-domain corpora. Since there is no standard test corpus is available, we created Hindi-Bengali educational test corpus through manual translation. We observed that the multilingual model outperformed all other models by achieving 34.8 BLEU and 0.466 TER points. We also conducted a human analysis of the multilingual model by taking a sample of 50 random sentences evaluated based on adequacy and fluency metrics by 4 evaluators. The model achieved average adequacy and fluency scores of 3.5 and 3.85, respectively.

9 Acknowledgement

Authors gratefully acknowledge the support from "Speech to Speech Machine Translation", Sponsored by PSA, Government of India.

References

Ahmed Abdelali, Francisco Guzman, Hassan Sajjad, and Stephan Vogel. 2014. The AMARA corpus: Building parallel language resources for the educational domain. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1856–1862, Reykjavik, Iceland. European Language Resources Association (ELRA).

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Maximiliana Behnke, Antonio Valerio Miceli Barone, Rico Sennrich, Vilelmini Sosoni, Thanasis Naskos, Eirini Takoulidou, Maria Stasimioti, Menno van Zaanen, Sheila Castilho, Federico Gaspari, Panayota Georgakopoulou, Valia Kordoni, Markus Egg, and

[25]These evaluators are the same who evaluated the quality of prepared clean in-domain corpus.

Katia Lida Kermanidis. 2018. Improving machine translation of educational content via crowdsourcing. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Denny Britz, Quoc Le, and Reid Pryzant. 2017. Effective domain mixing for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 118–126, Copenhagen, Denmark. Association for Computational Linguistics.

Chenhui Chu and Raj Dabre. 2019. Multilingual multi-domain adaptation approaches for neural machine translation. *arXiv preprint arXiv:1906.07978*.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of domain adaptation methods for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 385–391, Vancouver, Canada. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

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.

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*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 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, Prague, Czech Republic. Association for Computational Linguistics.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver. Association for Computational Linguistics.

Valia Kordoni, Antal van den Bosch, Katia Lida Kermanidis, Vilelmini Sosoni, Kostadin Cholakov, Iris Hendrickx, Matthias Huck, and Andy Way. 2016. Enhancing access to online education: Quality machine translation of MOOC content. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 16–22, Portorož, Slovenia. European Language Resources Association (ELRA).

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Santanu Pal, Sudip Kumar Naskar, Mihaela Vela, and Josef van Genabith. 2016. A neural network based approach to automatic post-editing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 281–286, Berlin, Germany. 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 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, Brussels, Belgium. Association for Computational Linguistics.

Gowtham Ramesh, Sumanth Doddapaneni, Aravinth Bheemaraj, Mayank Jobanputra, Raghavan AK, Ajitesh Sharma, Sujit Sahoo, Harshita Diddee, Divyanshu Kakwani, Navneet Kumar, et al. 2021. Samanantar: The largest publicly available parallel corpora collection for 11 indic languages. *arXiv preprint arXiv:2104.05596*.

Sukanta Sen, Kamal Kumar Gupta, Asif Ekbal, and Pushpak Bhattacharyya. 2018. IITP-MT at

WAT2018: Transformer-based multilingual indic-English neural machine translation system. In *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation: 5th Workshop on Asian Translation: 5th Workshop on Asian Translation*, Hong Kong. Association for Computational Linguistics.

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.

Dipti Misra Sharma, Asif Ekbal, Karunesh Arora, Sudip Kumar Naskar, Dipankar Ganguly, Sobha L, Radhika Mamidi, Sunita Arora, Pruthwik Mishra, and Vandan Mujadia, editors. 2020. *Proceedings of the 17th International Conference on Natural Language Processing (ICON): Adap-MT 2020 Shared Task*. NLP Association of India (NLPAI), Patna, India.

Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231, Cambridge, Massachusetts, USA. Association for Machine Translation in the Americas.

Vilelmini Sosoni, Katia Lida Kermanidis, Maria Stasimioti, Thanasis Naskos, Eirini Takoulidou, Menno van Zaanen, Sheila Castilho, Panayota Georgakopoulou, Valia Kordoni, and Markus Egg. 2018. Translation crowdsourcing: Creating a multilingual corpus of online educational content. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

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.

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.

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.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas. Association for Computational Linguistics.

Assessing Post-editing Effort in the English-Hindi Direction

Arafat Ahsan, Vandan Mujadia and **Dipti Misra Sharma**
IIIT, Hyderabad, India
{arafat.a, vandan.mu}@research.iiit.ac.in
dipti@iiit.ac.in

Abstract

We present findings from a first in-depth post-editing effort estimation study in the English-Hindi direction along multiple effort indicators. We conduct a controlled experiment involving professional translators, who complete assigned tasks alternately, in a translation from scratch and a post-edit condition. We find that post-editing reduces translation time (by 63%), utilizes fewer keystrokes (by 59%), and decreases the number of pauses (by 63%) when compared to translating from scratch. We further verify the quality of translations thus produced via a human evaluation task in which we do not detect any discernible quality differences.

1 Introduction

Translation workflows that are based on post-editing of Machine Translation output are being increasingly adopted in the industry (Gaspari et al., 2015). Gains that accrue from a post-editing based workflow, measured over multiple post-editing effort indicators, have been reported to be considerably significant by a number of previous studies over multiple language combinations (Plitt and Masselot, 2010; C. M. de Sousa et al., 2011; Green et al., 2013). But to extend post-editing beyond its current silos it is imperative to put new and less-studied language pairs under the lens to make a case for wider adoption via empirically backed evidence.[1]

Post-editing effort is often quantified across three different dimensions, each focusing in turn on a different aspect of post-editing behaviour (Krings, 2001). The dimensions studied are the following: *Temporal*– understood as the time taken to complete a translation task, often reported per segment or word; *Technical*– estimate of the physical labour of the translation activity, measured in terms of keystrokes logged or edit operations performed; and *Cognitive*– an indirect estimate of the extent of cognitive processes underlying the translation task, inferred from keylogging pause or eye-tracking data as it is not possible to observe these directly (Moorkens et al., 2015).

If it can be shown that post-editing machine translation output is temporally efficient, technically less laborious, and cognitively less demanding, then it can be recommended as the default workflow for large translation jobs. But this first calls for a comparison between machine translation based post-editing behaviour (henceforth PE) and unaided human translation from scratch (henceforth HT). Thus, the research questions that we pose are the following:

- Is post-editing effort as measured on temporal, technical and cognitive dimensions *lesser* in the PE condition than the HT condition for the English-Hindi direction?
- Is the quality of post-edited segments *equal to* translated segments as ascertained by human raters?
- Do automatic MT evaluation metrics *correlate* with PE effort indicators, when both measured at the segment level?

Most of this paper will focus on answering the first question in some detail. We are equally interested in the other two as well, but will only be presenting some initial results from a first attempt at tackling them.

The rest of this paper is organized as follows: Section 2 discusses some past studies including those that have previously studied the English-Hindi PE direction. In Section 3 we detail our experimental setup. Section 4 presents our results

[1] Gaspari et al. (2015)'s survey reveals a heavy skew towards English and other European language combinations.

and analysis and in Section 5 we draw our conclusions and sketch the outlines of our future work.

2 Related Work

We now take a more detailed look at some of the past efforts towards contrasting the two settings. Plitt and Masselot (2010) compared HT and PE when translating from English into 4 European languages (French, Italian, German, and Spanish) and reported an overall productivity gain of 74% which converted into time savings of 43%. They also observed a 70% reduction in keyboard time and 31% in pause time for the PE setting. C. M. de Sousa et al. (2011) also report PE to be 40% faster than HT in the English-Portuguese direction when translating movie subtitles.

Other studies however (Läubli et al., 2013), have reported more modest gains, with estimated time savings of 15–20% when translating between a European language pair (German-French) within a *realistic* translation environment.[2] Garcia (2011) also finds only marginal productivity gains when studying the English-Chinese pair and additionally reports an impact of directionality when source and target languages are switched.

All of these earlier experiments, however, were based on the output of Phrase based Statistical Machine Translation (PBSMT) systems. With Neural Machine Translation (NMT) and its subsequent iterations being the current state of the art and outperforming PBSMT (Bahdanau et al., 2014; Vaswani et al., 2017; Castilho et al., 2018), this shift in technology paradigm from PBSMT to NMT must then be addressed in post-editing studies as well.

Läubli et al. (2019) conduct such a study, this time utilizing the output of an NMT system to compare PE with HT in the German-French and German-Italian translation directions.[3] They report significant overall productivity gains, but with marked differences between the pairs: 59.74% for the former and only 9.26% for the latter. Another interesting comparison of HT, PBSMT, and NMT post-editing settings performed on a literary text (chapter from a novel) reports an increase in productivity by 36% for the NMT based setting over HT (Toral et al., 2018).

We have seen in previous studies that throughputs vary considerably depending on the language pair under the lens (Green et al., 2013; Läubli et al., 2019). We now discuss some earlier efforts that have included an Indian language in their experiment.

Shah et al. (2015) conducted an experiment where students post-edited parts of a specialized English language textbook on *bioelectromagnetism* into 7 languages, 3 of them being Indian languages including Hindi. They reported an increase in post-edit time by a factor of 3–5 when the target language was an Indian language. They put this down to greater terminological distance between English and Indian languages compared to other languages in their experiment. They do not study and compare against the HT condition, or report on technical or cognitive indicators.

Carl et al. (2016) also report results on Hindi (amongst 6 languages) comparing the HT and PE conditions. Their English-Hindi results are based on an existing multilingual translation database that contains experimental data around translators' activities in both conditions. They find in favour of the PE condition across all languages when measured on temporal indicators, but report translating into Hindi to be the slowest amongst the 6. They do not quantify average throughput gain or time savings.

Meetei et al. (2020) compare PE behaviour when translating from English into 3 Indian languages (Manipuri, Mizo and Hindi). They conduct light post-editing and report Hindi to be the fastest to post-edit amongst the three languages.[4] They ascribe it to the availability of relatively mature MT systems in the English-Hindi direction compared to Mizo and Manipuri, which are low-resource languages. They use student volunteers and do not investigate cognitive indicators.

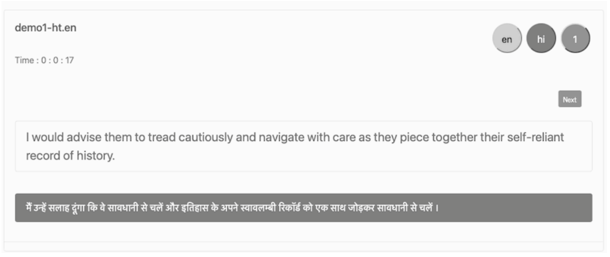
Ahmad et al. (2018) present an industry perspective and claim a 2–3 fold increase in productivity for users using their tools in combination with MT. However, they base this on longitudinal tracking of their users.

While all these past studies have certainly helped in providing insights into post-editing behaviour in Indian languages in general and in Hindi in particular, we sense a need for a more in-depth look along all PE effort indicators within at least one

[2]Experimental settings for these studies may deploy specialized interfaces for accurate measurements or make use of environments already familiar to professional translators.

[3]The HT condition is aided by a domain specific translation memory (TM).

[4]It is also often referred to as *good enough* translations and is lower than publishable quality translations.



Time : 0 : 0 : 17

I would advise them to tread cautiously and navigate with care as they piece together their self-reliant record of history.

मैं उन्हें सलाह दूंगा कि वे सावधानी से चलें और इतिहास के अपने स्वावलम्बी रिकॉर्ड को एक साथ जोड़कर सावधानी से चलें ।

Figure 1: Shows a snapshot of the translator workbench used in the study. Depicts an example in the PE condition pre-filled with an MT proposal; for the HT condition the text area is left blank.

Indian-language setting. With our current work, we seek to address this gap.

We see our contribution differentiated from previous related work as follows: *(i)* we present in-depth results from all three primary indicators of PE effort (temporal, technical and cognitive) for the English-Hindi direction; *(ii)* we account for per translator and per item variation with the use of mixed-effects models; *(iii)* we utilize professional translators in order to accurately gauge the impact of contrasting conditions; *(iv)* we conduct a human quality-rating exercise comparing target text produced in both conditions; *(v)* we present correlations of automatic MT metrics with PE effort indicators.

3 Experimental Setup

We conducted this experiment under a 2 (translation conditions) $\times$ 200 (source segments) mixed design. All subjects saw both factor levels (HT and PE), but only one combination for each level as having been exposed to a source segment in one condition would have affected their translation in the other. The study was conducted online over 5 consecutive days with 2 sessions per day. The sessions were not time bound. Subjects translated 2 files of 10 segments each in alternating conditions in each session.

All subjects participated in a warm-up translation exercise a week prior to the start of the actual task. This was done to establish familiarity with the interface used in the study. We chose to adapt an existing beta version of a web-based translation workbench by adding extensive keystroke logging features along with some other minor tweaks.[5] The UI itself was kept clean and uncluttered, serving one segment at a time to the translators. This meant that while translators had previous context of the text under translation, they could not navigate ahead for context. A timer was displayed once a translator navigated to each new segment. This was meant to prompt the translator to focus on the activity at hand. Figure 1 shows the workbench interface as seen by translators under the PE condition.

We instructed the participants to aim for publishable translation quality. They were free to conduct web searches and consult online or offline dictionaries, but were discouraged from spending too much time doing so.[6] It was deemed acceptable to transliterate any technical terms or terminology into Hindi if they could not find its translation even with the aid of resources available to them. However, they were strictly prohibited from consulting any online MT engines during the task. Subjects were encouraged to complete each task (consisting of 10 segments) in a single sitting without a break.

Previous studies, such as those discussed earlier, have noted the impact of a number of different variables (language pairs, MT paradigms, text domains, translation environments, translator competencies) on translation throughputs. This calls for not only careful experiment design, but also utilization of techniques that can help with the testing and inference of results. Green et al. (2013) utilized one of the first such designs for post-editing productivity studies and deployed mixed-effects models (Baayen et al., 2008) to account for inter-language, inter-subject, and inter-item variability.

Mixed-effects models are able to model this variability in two ways: *(i)* through random intercepts, that can account for the differences between translators seen in their differing throughputs (or differences between linguistic items due to the features inherent to them); *(ii)* and through a random slope that accounts for how different subjects may experience the change of condition differently. Accounting for these variabilities allows us to isolate the effect of condition, generalize our findings beyond our sample, and avoid the ”language-as-fixed-effect fallacy” (Clark, 1973).

In fitting our mixed-effects models we follow a methodology similar to the one described by Baayen et al. (2008) and followed by Green et al. (2013) and later Toral et al. (2018). Maximal models were fit when possible (Barr et al., 2013); in

[5]https://indictranslate.in/

[6]This was done as technical terminology related difficulties have previously been noted for this language direction (Shah et al., 2015).

Day	S1-T1	S1-T2	S2-T1	S2-T2
Day 1	20.20 (A1)	26.30 (A1)	26.80 (A1)	21.30 (A2)
Day 2	20.90 (A2)	22.30 (A2)	24.90 (A2)	23.20 (A3)
Day 3	26.40 (A4)	24.60 (A4)	26.30 (A4)	22.40 (A4)
Day 4	22.40 (A4)	20.30 (A4)	18.00 (A5)	18.70 (A5)
Day 5	22.50 (A5)	16.00 (A5)	20.30 (A5)	23.00 (A5)

Table 1: Average sentence lengths (in words) per session-task block as presented to translators. Also shown in parenthesis are the source articles used for each block.

case of convergence failure, a less complex model was fit by successively removing the random slopes of the by-subject and by-segment random effects component. Models thus obtained were compared via likelihood ratio tests. We also refit our final models after filtering data points with residuals deviating more than 2.5 standard deviations. This helps check for the influence of any atypical outliers (Baayen et al., 2008). We verify the residual plots for normality and homoscedasticity. We utilized the `lme4` package in R (Bates et al., 2015) for all mixed-effects models related analyses.

3.1 Data

We assembled a corpus of recent English language news articles from two distinct online sources. The choice of news as a domain was motivated by observations of terminology-related difficulties in more specialized domains, as reported by earlier studies (Shah et al., 2015). Each news article was segmented into sentences using the NLTK library and divided into *blocks* of 10 segments.[7] Only those blocks were used that fell within a *MEAN* ± *SD* of the corpus mean (Table 1). We prioritized the continuity of a news article across blocks when making *block* selections.[8] This methodology yielded a total of 200 unique source segments divided into 20 blocks of 10 segments each, spanning 5 different news articles: A1–A5. Conditions were counterbalanced to handle order effects.

3.2 Participants

The participants of our study are self-declared professional translators. We contacted a professional translation service provider to help assemble the pool.[9] A short questionnaire accompanied the registration form for the task. Of our participant pool of 10 subjects, 70% reported 2–5 years of experience translating in the English-Hindi direction, while 30% reported 0–2 years of experience. The same percentage breakdown was observed for a question related to previous post-editing experience. All subjects were paid the going market rates for the task regardless of the condition (PE, HT).

3.3 MT System

The English→Hindi MT engine used for the task is a transformer based neural machine translation system. This subword-based NMT system is trained on cleaned WAT 2021 [10] English-Hindi training corpus using the Opennmt-py toolkit (Klein et al., 2020). The system also utilizes forward and backward translations on the IndicCorp monolingual corpus to obtain synthetic data for training.[11] It uses subwords as the basic translation unit with 20,000 merge operations on both source and target languages. The system obtained a BLEU score of *35.46* on cleaned WAT 2021 English-Hindi test data.

4 Results and Discussion

4.1 Pre-processing

Once we processed the activity logs for all 10 subjects across all 200 segments, they yielded 2000 unique observations. We found that 7 of these items (all from the HT condition) did not contain a final translation, so we discarded those. We think that in these cases the subjects may have accidentally navigated to the next segment without having completed a translation. In the PE condition we found that one subject *P01* had not touched 68% of the MT outputs and had accepted them without modifications. This was almost 3 times the next highest proportion we detected across all other subjects. We decided to remove all data points generated by this subject. We were thus left with 1793 observations on which we base these results.

We calculated time per segment, source segment lengths (in words and characters), number of keystrokes (total, as well as those belonging to different categories: content, navigation and deletion), average pause duration, initial pause duration, and number of pauses. We also computed H-BLEU (Papineni et al., 2002), H-TER (Snover et al., 2006) and H-chrF (Popović, 2015) metrics on the post-edited segments.[12]

[7]https://www.nltk.org/api/nltk.tokenize.html

[8]In 2 cases out of 20 we had to skip the subsequent block, owing to short average sentence lengths of the blocks.

[9]http://www.ebhashasetu.com/

[10]http://lotus.kuee.kyoto-u.ac.jp/WAT/indic-multilingual/

[11]https://indicnlp.ai4bharat.org/corpora/

[12]H signifies that scores were computed using the reference generated in the PE condition by the same subject.

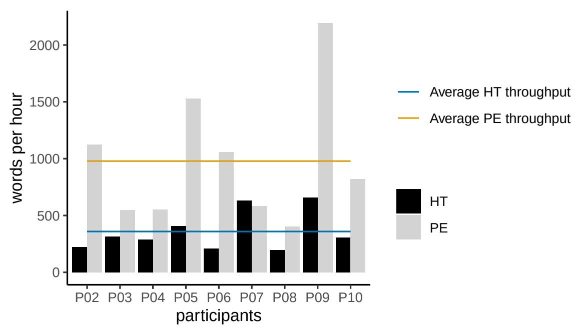


Figure 2: Individual translation throughputs in words per hour and average throughput in each contrasting condition.

Participant	Unedited (%)	Edited (%)	Productivity Gain (%)
P02	20	80	411
P03	3	97	73
P04	3	97	92
P05	18	82	275
P06	25	75	404
P07	0	100	-7
P08	2	98	105
P09	25	75	234
P10	16	84	169

Table 2: MT segments accepted without modifications and with modifications per subject along with individual productivity gain percentages.

Having assembled this data, we set out to answer the research questions posed earlier in Section 2.

4.2 Temporal Effort

We first present a view of temporal effort in terms of productivity measured as words per hour. We see productivity improvements in the PE condition across the board except for subject *P07*. Overall, this translates into a throughput increase from 359 words/hour to 979 words/hour. We thus observe an overall productivity gain of 172%, which amounts to 63% in time savings.

This is more than twice the 74% gain reported by (Plitt and Masselot, 2010) when studying European-language pairs and the 59.74% reported by (Läubli et al., 2019) recently. But we note that in the first case, the experiments were conducted on PBSMT outputs, and in the second, while NMT was used, the control condition was aided by a TM, thus pushing up the baseline throughputs. With this context in mind, the average productivity gain seen in our study does not appear to be unrealistic.

Figure 2 shows a comparison of individual throughputs in contrasting task conditions along with means aggregated for the two conditions. We also note a great variation in productivity gain amongst subjects ranging from -7% to 410%.

Table 2 helps interpret this further. We contrast the number of unedited and edited MT proposals per subject and their individual productivity gain percentages. It follows that higher the acceptance of MT proposals without modifications by a subject, greater the gain in individual productivity. While this may point to high quality MT output, it also demands a closer scrutiny of the quality of translations generated in each condition. We address this in Section 4.5.

We now report the mixed-effects regression results. Plotting temporal data showed a right-skewed distribution, so we log transform all time data before proceeding further. As our goal is to predict translation time and establish the significance of conditions, we fit a linear mixed-effects regression model with two fixed-effect predictors (condition and segment length) and two random-effect predictors (subjects and segments), where on the subject predictor we also include a random-slope for task condition.

In the final model, we observe a significant main effect for both segment length as well as translation condition.[13] Temporal effort significantly increases with segment length, but decreases for the PE condition. Table 3 shows the significance levels and direction for each predictor in our final models across all PE effort dimensions that we study.

4.3 Technical Effort

We measure technical effort as the number of keystrokes used to generate the target text. We normalize it per source segment character. Figure 3 shows 1.33 keystrokes used per source character in the HT condition and 0.54 keystrokes in the PE condition, amounting to an effort reduction of 59%. Contrast this with the 23% reduction reported by Toral et al. (2018) when post-editing a literary text. Again, except for subject *P07* all participants show reduced effort in the PE condition.

We also classified each keystroke based on the type of the keystroke logged. We classify these into content, navigation, and deletion categories and report the percentage breakdown of the total into these categories in Table 4. We observe higher navigation and deletion operations in the PE condition (28% and 26%) than the HT condition (8% and 14%), while content keystrokes register a higher percentage in HT (77%) compared to PE (46%).

[13]We utilize the *lmerTest* package that extends results with p-values for models built with *lme4*.

Predictor	Temporal	Technical	Cognitive		
			number	average duration	initial duration
Segment length	$\uparrow^{***}$	$\uparrow^{***}$	$\uparrow^{***}$	$\uparrow^{***}$	$\uparrow^{***}$
Condition (PE vs. HT)	$\downarrow^{***}$	$\downarrow^{***}$	$\downarrow^{***}$	–	–

Significance levels: —$(p > 0.1)$, $(p < 0.1)$, $^{*}(p < 0.05)$, $^{**}(p < 0.01)$, $^{***}(p < 0.001)$.
Direction: ($\uparrow\downarrow$) arrows depict whether the predictor has a negative or positive correlation with the dependent variable.

Table 3: Significance levels of predictors in our final models across all modeled PE effort dimensions.

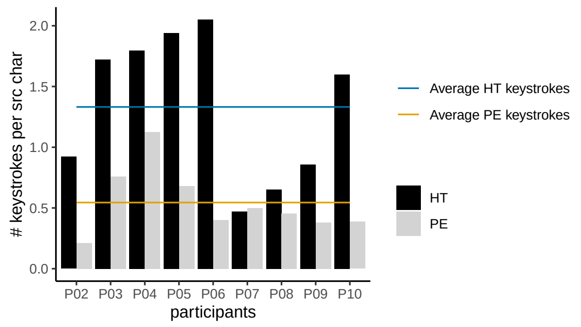


Figure 3: Technical effort estimated as number of keystrokes needed to generate target text per source character.

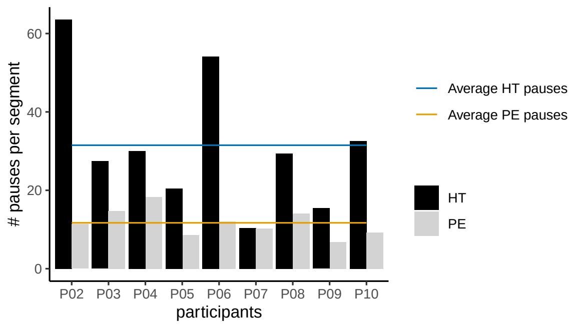


Figure 4: Cognitive effort estimated as average number of pauses per source segment.

Subject *P08* is an interesting case as they register the highest number of delete operations (they have high navigation numbers too) in either condition amongst all participants. This could point to frequent revisions made on the text.

As number of keystrokes is expressed as counts, we fit a Poisson generalized linear mixed-effects model to predict technical effort. We follow the same methodology as described in the previous section. We again find a significant main effect both for segment length as well as translation condition (Table 3), similar to what we saw for the temporal dimension earlier. Technical effort increases with increase in segment length, and decreases for the change in condition to PE.

4.4 Cognitive Effort

Post-editing effort estimation studies based on eye-tracking data use *fixations* as a proxy to estimate cognitive load; the idea being that greater the number and duration of fixations, greater the cognitive load (O'Brien, 2011). In the absence of eye-tracking data, the use of *pauses* as a proxy for cognitive load is also well established (O'Brien, 2006). We report on three such cognitive indicators: number of pauses, pause duration, and initial pause duration. Findings related to the first two have been reported in previous post-editing literature (Green et al., 2013; Toral et al., 2018).[14] The third (initial pause duration), we introduce in order to gauge differences in reaction times from when a subject first navigates to a new segment displayed in either condition to their first action on it.

We calculate the time difference between two subsequent key events and consider all observations above *1000ms* to be pauses following (O'Brien, 2006; Koehn, 2009).

Figure 4 shows the differences in the frequency of pauses for each subject in the two conditions. We notice a reduction of 63% in the PE condition from 31 pauses per segment in HT to 12 pauses per segment in PE. This points to a much reduced cognitive load when post-editing.

However, a similar exercise on pause duration data reveals an increase of approximately 12% in the PE condition compared to the HT condition (Figure 5). Although, it is not significant, this is in line with findings reported previously comparing these two specific cognitive indicators (Green et al., 2013).

We finally compare initial pause duration between PE and HT. We expect this initial load to be higher for the PE condition given that there are two

[14]There is also an indicator reported as pause ratio which we eschew in favour of initial pause time.

Participant	HT (%)			PE (%)		
	Content	Navigation	Deletion	Content	Navigation	Deletion
$P02$	81	9	11	41	32	27
$P03$	94	1	5	58	8	34
$P04$	86	9	5	48	41	11
$P05$	85	5	10	56	30	14
$P06$	90	1	9	54	28	18
$P07$	66	11	23	56	12	32
$P08$	24	25	51	10	27	63
$P09$	78	12	10	35	52	13
$P10$	94	1	5	59	19	22
$Mean \pm SD$	77.55 ± 21.80	8.15 ± 7.89	14.30 ± 14.63	46.52 ± 15.91	27.69 ± 13.65	25.8 ± 16.1

Table 4: Types of keystrokes generated by subjects in each condition.

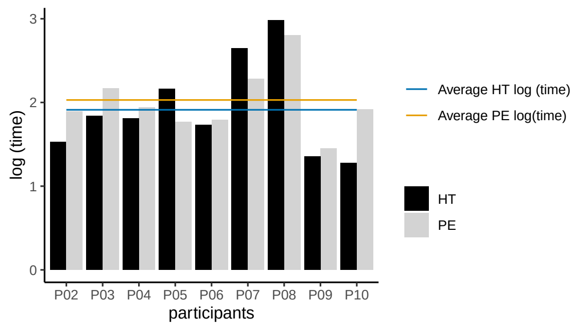


Figure 5: Cognitive effort estimated as average pause duration per source segment.

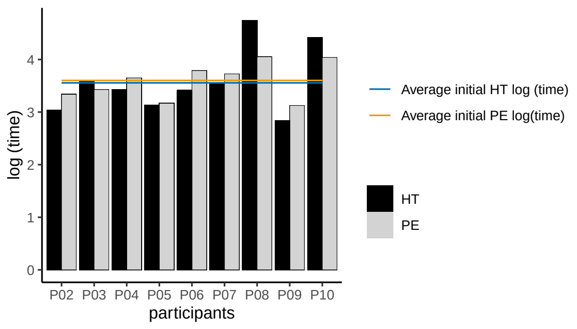


Figure 6: Cognitive effort estimated as average initial pause duration per source segment.

segments displayed to the subject in this condition: the source segment and the MT proposal, which have to be read and comprehended before starting the post-editing activity. This seems to hold, but not significantly, as we see only a small increase of about 5% for the PE condition (Figure 6). One explanation could be that in the PE condition, the MT proposal in spite of registering a higher cognitive load initially also later acts as a helpful prompt for the subject. An eye-tracking based experiment might prove useful in teasing apart these two opposite effects.

When comparing the means[15] for pause duration and initial pause duration we find pause duration (6.77s for HT and 7.71s for PE) to be considerably lower than initial pause duration (35.02s for HT and 36.67s for PE). The translator therefore, takes a longer initial pause to start formulating a response, but once they start the activity, they take considerably shorter pauses.

We go on to fit three mixed-effects models to validate these findings. A Poisson generalized mixed-effects model to estimate pause counts finds significant main effects for segment length and condition. Cognitive effort (measured as count of pauses) increases with segment length and decreases for the change in condition to PE.

The other two linear mixed-effects models fit to predict average pause duration, and initial pause duration find a significant effect only for segment length and not for condition (Table 3). This shows that while cognitive effort certainly increases with segment length, the change in condition to PE, does not have a discernible effect on cognitive effort, when measured by the average and initial time duration indicators.

4.5 Quality Judgements

To evaluate whether the quality of texts created in the PE condition matched those created in the HT condition, we conducted a human judgement based pairwise ranking task (Callison-Burch et al., 2011) on a small sample. We randomly sampled 3 target segments per condition from each subject. For each target text thus obtained, we paired it with another random sample after constraining on condition. We thus obtained 60 HT-PE pairs for evaluation. As we discovered issues (discussed earlier in Section 4.1) with subject *P01*'s data after the quality eval-

[15]After transforming back from log scale. Also, note that pause duration does not include initial pause duration as a component, and is the duration of pauses after post-editing starts.

Evaluator	PE vs. HT		
	Win	Loss	Tie
$E1$	18	14	15
$E2$	27	15	5
$E3$	7	7	33
$E4$	14	11	22
$E5$	13	11	23

Table 5: Pairwise quality judgements on sampled target texts reported as win, loss, and ties for PE against HT.

MT Metric	PE Indicator		
	temporal	technical	cognitive (# pauses)
$(H)BLEU$	r = $-.56,^{***}$	r = $-.71,^{***}$	r = $-.48,^{***}$
$(H)TER$	r = $+.54,^{***}$	r = $+.70,^{***}$	r = $+.49,^{***}$
$(H)chrF$	r = $-.56,^{***}$	r = $-.73,^{***}$	r = $-.49,^{***}$

Note: All coefficients for r(898). For TER lower is better hence the positive correlation. The other two cognitive indicators (average and initial pause duration) did not show any correlation with any of the metrics – coefficients were close to 0.

Table 6: Correlations of PE Effort indicators with automatic MT metrics.

uation exercise had already been completed, we removed any pairs that had a segment translated by the subject. We report our results on this filtered set that consists of 47 pairs. Five evaluators were asked to judge each pair. Ties were allowed.

Table 5 shows the judgements from evaluators represented as win-loss statistics on the PE condition. We notice a high number of ties and a slight preference for the PE condition. However, the preference does not test to be significant on a sign test ignoring ties (p-value = 0.08). We conclude that translating in either condition produces similar quality target segments. However, we realise that the sample size was quite small compared to the number of possible combinations across all participants. We hope to conduct a more thorough review of quality in future.

4.6 Automatic Quality Metrics

Finally, we investigate the correlations of some popular automatic MT evaluation metrics with the post-editing effort indicators reported so far in this study. We generated metric scores by comparing the MT proposal against its post-edited reference. We calculate scores for H-(BLEU, TER, and chrF).

Table 6 shows moderate correlations for all 3 MT metrics on the temporal indicator, similar to what Tatsumi (2009) also reported for this indicator. Correlations then get stronger on the technical indicator and then fade for the cognitive indicator.

We believe this may be because cognitive effort is the only one out of the three PE dimensions we studied that is not directly observed (instead, inferred from pause frequency and pause duration data), whereas the technical and temporal indicators can be measured more directly. This is similar to findings previously reported by Moorkens et al. (2015) who note a similar correlation trend across the three PE effort dimensions. The technical effort indicator appears to be the one most strongly correlated with automatic metrics.

The other two cognitive indicators (average and initial pause duration), which did not test significant as per our mixed-effects models, also do not show any correlation with any of the MT metrics – coefficients obtained were close to 0. We omit reporting them in Table 6 due to space constraints.

5 Conclusion

We conducted a post-editing effort assessment study and presented detailed analysis of effort indicators along the temporal, technical and cognitive dimensions. We observed that in the temporal dimension, post-editing reduced translation time by 63%; in the technical dimension it reduced number of key strokes by 59%; and in the cognitive dimension, it reduced the frequency of pauses by 63%. However, it increased average pause duration by 12% and average initial pause duration by 5%.

We then compared the quality of translations generated in each condition and found them to be similar.

And finally, we detected moderate to strong correlations for 3 automatic MT evaluation metrics across all PE effort indicators, with technical effort most strongly correlating with automatic MT metrics.

The last two observations regarding human quality judgement, and MT metrics and their correlations demand a closer look, which was not possible owing to time and space constraints. We expect to undertake this as part of our future work. We also propose to extend this study by including a third condition in future, either as an additional MT engine to check if MT quality differences show up in PE effort indicators (Toral et al., 2018), or by the use of translation aids (TM) to gauge their impact in a similar manner (Läubli et al., 2013, 2019).

We also intend to study other language pairs, especially those within the multilingual Indian context.

References

Rashid Ahmad, Priyank Gupta, Nagaraju Vuppala, Sanket Kumar Pathak, Ashutosh Kumar, Gagan Soni, Sravan Kumar, Manish Shrivastava, Avinash K Singh, Arbind K Gangwar, et al. 2018. Transzaar: Empowers human translators. In *2018 18th International Conference on Computational Science and Applications (ICCSA)*, pages 1–8. IEEE.

R Harald Baayen, Douglas J Davidson, and Douglas M Bates. 2008. Mixed-effects modeling with crossed random effects for subjects and items. *Journal of memory and language*, 59(4):390–412.

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

Dale J Barr, Roger Levy, Christoph Scheepers, and Harry J Tily. 2013. Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of memory and language*, 68(3):255–278.

Douglas Bates, Martin Machler, Ben Bolker, and Steve Walker. 2015. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Omar Zaidan. 2011. Findings of the 2011 workshop on statistical machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 22–64.

Michael Carl, Akiko Aizawa, and Masaru Yamada. 2016. English-to-japanese translation vs. dictation vs. post-editing: Comparing translation modes in a multilingual setting. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4024–4031.

Sheila Castilho, Joss Moorkens, Federico Gaspari, Rico Sennrich, Andy Way, and Panayota Georgakopoulou. 2018. Evaluating mt for massive open online courses. *Machine translation*, 32(3):255–278.

Herbert H Clark. 1973. The language-as-fixed-effect fallacy: A critique of language statistics in psychological research. *Journal of verbal learning and verbal behavior*, 12(4):335–359.

Ignacio Garcia. 2011. Translating by post-editing: is it the way forward? *Machine Translation*, 25(3):217–237.

Federico Gaspari, Hala Almaghout, and Stephen Doherty. 2015. A survey of machine translation competences: Insights for translation technology educators and practitioners. *Perspectives*, 23(3):333–358.

Spence Green, Jeffrey Heer, and Christopher D Manning. 2013. The efficacy of human post-editing for language translation. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 439–448.

Guillaume Klein, François Hernandez, Vincent Nguyen, and Jean Senellart. 2020. The opennmt neural machine translation toolkit: 2020 edition. In *Proceedings of the 14th Conference of the Association for Machine Translation in the Americas (AMTA 2020)*, pages 102–109.

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

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

Samuel Läubli, Chantal Amrhein, Patrick Düggelin, Beatriz Gonzalez, Alena Zwahlen, and Martin Volk. 2019. Post-editing productivity with neural machine translation: an empirical assessment of speed and quality in the banking and finance domain. *arXiv preprint arXiv:1906.01685*.

Samuel Läubli, Mark Fishel, Gary Massey, Maureen Ehrensberger-Dow, Martin Volk, Sharon O'Brien, Michel Simard, and Lucia Specia. 2013. Assessing post-editing efficiency in a realistic translation environment.

Loitongbam Sanayai Meetei, Thoudam Doren Singh, Sivaji Bandyopadhyay, Mihaela Vela, and Josef van Genabith. 2020. English to manipuri and mizo post-editing effort and its impact on low resource machine translation. In *Proceedings of the 17th International Conference on Natural Language Processing (ICON)*, pages 50–59.

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.

Sharon O'Brien. 2006. Pauses as indicators of cognitive effort in post-editing machine translation output. *Across Languages and Cultures*, 7(1):1–21.

Sharon O'Brien. 2011. Towards predicting post-editing productivity. *Machine translation*, 25(3):197–215.

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.

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 Popović. 2015. chrf: character n-gram f-score for automatic mt evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395.

Ritesh Shah, Christian Boitet, Pushpak Bhattacharyya, Mithun Padmakumar, Leonardo Zilio, Ruslan Kalitvianski, Mohammad Nasiruddin, Mutsuko Tomokiyo, and Sandra Milena Castellanos Páez. 2015. Post-editing a chapter of a specialized textbook into 7 languages: importance of terminological proximity with english for productivity. In *Proceedings of the 12th International Conference on Natural Language Processing*, pages 325–332.

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 the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231.

Sheila C. M. de Sousa, Wilker Aziz, and Lucia Specia. 2011. Assessing the post-editing effort for automatic and semi-automatic translations of DVD subtitles. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pages 97–103, Hissar, Bulgaria. Association for Computational Linguistics.

Midori Tatsumi. 2009. Correlation between automatic evaluation metric scores, post-editing speed, and some other factors. *The Twelfth Machine Translation Summit (MT-Summit XII)*, pages 332–339.

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.

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.

An Experiment on Speech-to-Text Translation Systems for Manipuri to English on Low Resource Setting

Loitongbam Sanayai Meetei[1], **Laishram Rahul**[2], **Alok Singh**[1], **Salam Michael Singh**[1], **Thoudam Doren Singh**[1], and **Sivaji Bandyopadhyay**[1]

[1]Centre for Natural Language Processing (CNLP) & Dept. of CSE, NIT Silchar, India
[2]Dept. of CSE, SIT, Tumkur
{loisanayai,laishramrahulib,alok.rawat478,salammichaelcse,
thoudam.doren,sivaji.cse.ju}@gmail.com

Abstract

In this paper, we report the experimental findings of building Speech-to-Text translation systems for Manipuri→English on low resource setting which is first of its kind in this language pair. For this purpose, a new dataset consisting of a Manipuri-English parallel corpus along with the corresponding audio version of the Manipuri text is built. Based on this dataset, a benchmark evaluation is reported for the Manipuri→English Speech-to-Text translation using two approaches: 1) a pipeline model consisting of ASR (Automatic Speech Recognition) and Machine translation, and 2) an end-to-end Speech-to-Text translation. Gaussian Mixture Model-Hidden Markov Model (GMM-HMM) and Time delay neural network (TDNN) Acoustic models are used to build two different pipeline systems using a shared MT system. Experimental result shows that the TDNN model outperforms GMM-HMM model significantly by a margin of 2.53% WER. However, their evaluation of Speech-to-Text translation differs by a small margin of 0.1 BLEU. Both the pipeline translation models outperform the end-to-end translation model by a margin of 2.6 BLEU score.

1 Introduction

In recent times, the advance in machine translation (MT) systems research jumped from textual modality to multi modality. The success of the several machine translation system for major languages based on statistical and neural approaches shed light towards building better translations systems for low resource languages as well. Of these, the statistical machine translation (SMT) (Koehn et al., 2003) and neural machine translation (NMT) models (Cho et al., 2014) started its journey from the traditional text-to-text translation which further expanded to the use of multiple modalities (Huang et al., 2016; Caglayan et al., 2016; Meetei et al., 2019; Gain et al., 2021) in the translation task. The usage of multiple modalities in MT uncovers new avenues for MT researchers. MT tasks where multiple modalities are utilized include using multiple-input modalities, for example, incorporating visual and text modalities (Meetei et al., 2021; Singh et al., 2021), translation between different input and output modalities such as Speech-to-Text translation (Ney, 1999; Weiss et al., 2017), etc. With these various methodologies of MT, the main goal is to obtain the most key information in a modality in generating the optimal sentence translation.

The Speech-to-Text (S2T) translation is the translation of a speech in a source language to a target language text. The Speech-to-Text translation task can be broadly addressed using two approaches: 1) with a pipeline strategy, which separates the different modalities into modality conversion, i.e., ASR, followed by text-to-text MT. 2) end-to-end (E2E) translation where the target text is directly generated from the speech in the source language. The Speech-to-Text (S2T) can find its application in our daily life by creating an ease form of communication for individuals with physical disabilities. It can also be used in reducing the turnaround of quick documentation, generating subtitles, etc.

Despite the fact that researchers are pushing the frontiers in machine translation and improving their capabilities, most of the work is focused on well-studied languages while work on low resource languages such as Manipuri is falling behind. Manipuri (also known as Meiteilon) is the official language of Manipur, a northeastern state of India. Manipuri is an extremely low resource language with a limited dataset available for the NLP (Natural Language Processing) tasks which is one of the primary reasons that hindered the development of NLP systems for the language.

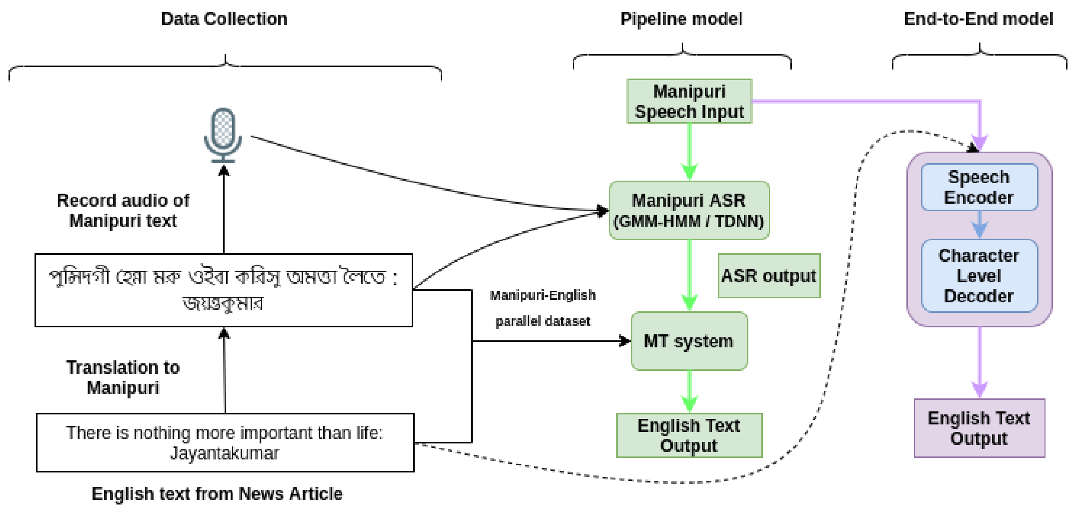

Data Collection
Pipeline model
End-to-End model
Manipuri Speech Input
Record audio of Manipuri text
Manipuri ASR (GMM-HMM / TDNN)
ASR output
পুন্সিদগী হেন্না মরু ওইবা করিসু অমত্তা লৈতে : জয়ন্তকুমার
Manipuri-English parallel dataset
MT system
Translation to Manipuri
English Text Output
There is nothing more important than life: Jayantakumar
English text from News Article
Speech Encoder
Character Level Decoder
English Text Output

Figure 1: Manipuri→English S2T translation models

This work aims to promote Speech-to-Text translation of an extremely low resource language by presenting a benchmark evaluation on a manually collected speech dataset. This work makes the following contributions:

- We build the first Manipuri→English S2T translation dataset.
- Comparison between a pipeline and end-to-end S2T translation model on the collected corpus is reported as the benchmark evaluation.

The rest of this paper is presented as follows: The prior relevant research is discussed in Section 2, followed by the framework of our model in Section 3. Section 4 and Section 5 explain the setup of our system and analysis of our results. The conclusion and future work are summarized in Section 6.

2 Related Works

Early attempts to address S2T translation follows a pipeline approach of two independent models: ASR and MT systems (Ney, 1999; Matusov et al., 2005). The approach utilized the hypothesis of ASR as an input to the MT model to generate the target-language text. Initial work on direct Speech-to-Text translation includes (Bérard et al., 2016; Duong et al., 2016; Bansal et al., 2017). Using a small French-English synthetic dataset from 7 speakers, Bérard et al. (2016) carried out an end-to-end S2T translation. The author reported that their system to be capable of generalizing to a new speaker effectively. Bansal et al. (2018) carried out an end-to-end S2T translation in low resource settings by training with smaller subsets of 160 hours labeled data. The author reported a BLEU score of 5.3 and 29.4 when trained with 20 hours and 160 hours, respectively.

Some of the work in the development of speech technology for the Manipuri language includes Rahul et al. (2013); Patel et al. (2018); Devi et al. (2021). Patel et al. (2018) reported a WER of 19.28% on a GMM-HMM and WER of 13.57% on a Deep Neural Network-HMM (DNN-HMM) acoustic model systems. The speech corpus used in the experiment comprised around 61 hours. Works on MT for Manipuri-English language pair are reported using various techniques such as Example-based MT (Singh and Bandyopadhyay, 2010a), SMT (Singh and Bandyopadhyay, 2010b; Singh, 2013), and unsupervised NMT (Singh and Singh, 2020). In a comparative study of SMT and NMT systems on the Manipuri-English language pair, the authors (Rahul et al., 2021; Singh and Singh, 2021) reported NMT system to perform better than the SMT system. To date, there is no work in S2T translation for Manipuri-English language pair. In order to fill this gap, a Manipuri-English S2T translation is developed using a small dataset in our work.

3 Methodologies

Figure 1 illustrates the methodology of our work. As the initial step of our work, English text dataset is collected from news articles, which is translated to Manipuri language. In the next step, speech is recorded for the Manipuri text. The overall collected dataset is then used to train the pipeline and End-to-End S2T translation models.

3.1 Language Resources

To build the dataset for our experiment, we collected news articles reported in English from a local daily newspaper[1]. The collected English text is machine translated to Manipuri followed by manual post-editing of the MT output and training the MT system with the incremental approach (Meetei et al., 2020). Following the development of the parallel dataset, speech is recorded for each of the Manipuri sentences by the native speakers of Manipur. The total number of participants for speech records is five: one male speaker and four female speakers. There is no overlapping of utterances among the participants. The recorded speech is post-processed, where the quality of speech records are verified manually. Any invalid speech found is rerecorded to collect quality speech records for the experiment. The overall collected dataset comprises of:

- 3500 Manipuri-English parallel text datasets, and
- around 5 hrs 30 minutes of speech record of the Manipuri text.

3.2 Speech Feature Extraction

For any Speech-to-Text system, extracting the audio signal components that can be used to determine linguistic content is important. Mel-frequency cepstral coefficients (MFCCs), the most popular, extensively utilized cepstral feature for ASR, is used as the audio feature for the ASR system and the E2E Speech-to-Text translation system.

3.3 Pipeline translation model: ASR and MT

Our pipeline S2T translation model consists of two independent models:

- Automatic Speech Recognition, and
- Neural Machine Translation (NMT)

In our work, we built two separate pipeline systems using GMM-HMM and TDNN Acoustic models, which is followed by a shared NMT system. The ASR output is fed to the NMT system to generate the target language.

[1]Imphal Free Press `https://www.ifp.co.in/`

3.3.1 Automatic Speech Recognition (ASR)

The objective of an automatic speech recognition system is to predict the most likely discrete symbol sequence from a given input acoustic speech vector O, out of all valid sequences in a target language T. Taking input speech sequence as a set of observation O= $(o_1, o_2, ...o_n)$ and the symbol to be predicted represented by S = $(s_1, s_2, ...s_n)$, the aim of the ASR model is:

$$\hat{S} = argmaxP(O|S)P(S). \tag{1}$$

where P(S) is the prior probability for the sequence S, and the observation likelihood, P(O|S) is the likelihood of the acoustic input sequence O given the sequence S, computed using HMM.

The acoustic model based on deep neural networks is trained with time delay neural network, TDNN (Peddinti et al., 2015).

3.3.2 Neural Machine Translation (NMT)

A Neural Machine Translation (NMT) is built for the MT system in the pipeline model. For a source sentence, $\mathbf{S} = \{s_1, \ldots, s_n\}$, NMT, an encoder-decoder sequence-to-sequence technique, jointly models the conditional probability $p(\mathbf{T}|\mathbf{S})$ to translate a target sequence, $\mathbf{T} = \{t_1, \ldots, t_m\}$.

Following the attention mechanism (Bahdanau et al., 2014; Luong et al., 2015), a bi-LSTM (Sutskever et al., 2014) is used as an encoder. At time step t, the encoder state is represented by the concatenation of the forward hidden state, $\vec{h_i}$, and backward hidden state, $\overleftarrow{h_i}$. As each word in the output sequence is decoded, the attention mechanism learns where to focus attention on the input sequence.

3.4 End-to-End S2T translation model

Our end-to-end S2T translation model follows Bérard et al. (2018) architecture, an attentive encoder-decoder model. The speech encoder takes audio features, X= $(x_1, x_2, ..., x_{T_x}) \in \mathbb{R}^{T_x \times N}$ as an input sequence. The audio features are fed into two non-linear $(tanh)$ layers, which generate $\acute{N}$ size features. The new feature sequence length is reduced by a factor of 4 using two 2D convolutional layers with stride (2; 2), which is then passed to a three stacked bidirectional LSTMs (Schuster and Paliwal, 1997). The decoder generates target-language sequences at the character level. The character-level decoder is composed of a conditional LSTM with the global attention mechanism (Bahdanau et al., 2014).

	sentences	duration (in min)
train	3300	~314
dev	100	~9
test	100	~8

Table 1: Manipuri→English Speech-to-Text translation dataset setup

4 Experimental Setup

In this section, we present the different Speech-to-Text translation experiments conducted, including the dataset and experimental setup.

The training, development, and test data sets for Manipuri→English S2T translation models are summarized in Table 1.

4.1 Pipeline S2T Translation Models

The system set up of independent ASR and MT systems of pipeline S2T translation model are as follows:

4.1.1 ASR systems

The transcript of the Manipuri text is written in Bengali script. Words in Manipuri have exact grapheme-to-phoneme mapping. A grapheme-to-phoneme list for the Manipuri ASR system is prepared by using the Bengali to Roman script transliteration module of (Meetei et al., 2021). The acoustic features fed to the GMM-HMM model consists of 13-dimensional MFCC, and 3-dimensional pitch features for speaker adaptation, namely Probability of Voicing (POV)-weighted mean subtraction over 1.5 second windows, Normalized Cross Correlation Function (NCCF)-derived POV feature, and delta pitch calculated on raw log pitch. While TDNN acoustic models are trained using 40-dimensional MFCC with 100-dimensional i-vectors and 3-dimensional pitch features. We utilized a 3-gram model trained with SRILM (Stolcke, 2002) for decoding. The ASR systems are built using the Kaldi toolkit (Povey et al., 2011).

4.1.2 NMT systems

Two NMT systems are trained using different dataset set up:

- NMT_{in}: NMT model trained with the in-domian dataset (Table 1).
- NMT_g: NMT model trained by combining the in-domain and additional parallel Manipuri-English text dataset. The additional dataset is acquired from TDIL[2], data scrapped from vikaspedia [3] which are then manually aligned and the work from (Meetei et al., 2020). Overall, the domain of the dataset is from tourism, agriculture, medical and news articles. The total training dataset size is 23126 (3300 in-domain and 19823 additional parallel sentences).

As an encoder, a two-layer bi-LSTM with 512 hidden units is used, and the batch size is set to 32. With a learning rate of 0.001 and Adam optimizer (Kingma and Ba, 2014), we train the system utilizing early stopping, where training is halted if a model does not progress on the validation set for more than 15 epochs.

4.2 End-to-End S2T Translation Model

End-to-End S2T translation models are implemented in `PyTorch` (Paszke et al., 2019) with `fair-seq` toolkit[4]. We utilize "`T-Sm`" architecture (Wang et al., 2020) with default hyper-parameters and train with Adam optimizer and a learning rate of 0.002. Early stopping is used to halt the training when the system does not improve for 15 epochs on the development set.

4.3 Evaluation Metrics

The word error rate (WER), which is the ratio of word insertion, deletion, and substitution errors in a transcript to the total number of uttered words, is used to evaluate our ASR systems. The final hypothesis of S2T are evaluated with BLEU (Papineni et al., 2002). BLEU is a precision-based automatic metric used to evaluate the quality of machine-translated text.

5 Results and Analysis

In this section, we illustrate the results of our Manipuri→English pipeline and end-to-end S2T translation models. Along with the automatic metric evaluation, we carried out an in-depth qualitative analysis and human evaluation of our translation systems.

5.1 Automatic Metrics based Evaluation

The ASR systems are evaluated in terms of word error rate (WER), and the final hypothesis of translation from the pipeline and end-

[2]`https://tdil-dc.in/`
[3]`https://vikaspedia.in/`
[4]`https://github.com/pytorch/fairseq`

	Acoustic Model	WER	MT	BLEU	Translation Model
Pipeline	GMM-HMM	27.69	NMT_{in}	6.1	PipeHmmIN
			NMT_g	4.6	PipeHmmG
	TDNN	**25.16**	NMT_{in}	**6.2**	PipeTdnnIN
			NMT_g	4.1	PipeTdnnG
E2E	-	-	-	3.6	E2E

Table 2: Manipuri→English Speech-to-Text translation results

.

transcript1	অহুমশুবা কৱাথা কুম্মৈ হৌদোকখ্রে *Third Kwatha Festival inaugurated*
GMM-HMM	৩ শুবা কৱাথা কুম্মৈ হৌদোকখ্রে
TDNN	অহুমশুবা কৱাথা কুম্মৈ হৌদোকখ্রে
transcript2	দ্রাইভরশিংগী য়ুনিয়ননা বন্দ য়েথোকখ্রে *Drivers union suspends bandh*
GMM-HMM	দ্রাইভরশিংগী য়ুনিয়ননা বন্দ য়েথোকখ্রে
TDNN	দ্রাইভরশিংগী য়ুনিয়ননা ভাবন য়েথোকখ্রে
transcript3	ওল জিরিবাম রোড ত্রান্সপোর্ট দ্রাইভরস য়ুনিয়ননা বন্দ য়েথোকখ্রে *All Jiribam Road Transport Drivers Union suspends bandh*
GMM-HMM	ওল জিরিবাম ভোট ট্রান্সপোর্ট দ্রাইভর য়ুনিয়ননা বন্দ য়েথোকখ্রে
TDNN	ওল জিরিবাম ভোট ট্রান্সপোর্ট দ্রাইভর য়ুনিয়ননা বাল য়েথোকখ্রে
transcript4	লৈবাক ৩১ লানলবা মতুংদা ইন্দিয়ান বাইকরশিং ইম্ফাল য়ৌরকখ্রে *Indian bikers reach Imphal after crossing 31 countries*
GMM-HMM	লৈবা ৩১ লানলবা মতুংদা ইন্দিয়ান বাইকরশিং ইম্ফাল য়ৌরকখ্রে
TDNN	লৈবা ৩১ লানলবা মতুংদা ইন্দিয়ান বাইকরশিং ইমফাল য়ৌরকখ্রে

Table 3: Sample input-output of Manipuri Automatic Speech Recognition systems

to-end systems is measured in terms of BLEU score using SacreBLEU (Post, 2018). Table 2 shows the automatic evaluation score of the ASR (GMM-HMM and TDNN) output and the translation output. The signature of the SacreBLEU is : $BLEU + case.mixed + numrefs.1 + smooth.exp + tok.13a + version.1.5.1$.

- **ASR:** TDNN model outperforms the GMM-HMM model significantly by achieving an improvement of 2.53% WER.
- **Translation**: The pipeline model with TDNN ASR and NMT_{in} achieve the highest BLEU score.

From the results in Table 2, it is observed that the evaluation of the target language translations from the output of the ASR systems using a shared NMT system differ by a small margin. The TDNN pipeline model achieve a 0.1 to 0.5 BLEU score more than the GMM-HMM pipeline model.

Comparing the evaluation scores of the translation hypothesis from the pipeline and End-to-End models, it is clear that the pipeline models outperforms the End-to-End model significantly by a margin of 2.6 BLEU score. The result also shows that the usage of additional out of domain data where the size of the dataset is substantially larger than the in-domain dataset size has negative effect on the BLEU score. A likely cause is the use of development and test dataset from the in-domain dataset.

5.2 Qualitative Analysis of Manipuri ASR Systems

Table 3 shows some sample input-output of Manipuri ASR systems where we analyse the robustness of the systems on selected words in the reference transcript highlighted in green.

In **transcript1**, "অহুমশুবা" (~ "ahumsuba" meaning *third*) is generated in its numerical format "৩ শুবা" (~ "3 suba" meaning 3^{rd}) by GMM-HMM ASR system while the TDNN ASR system generate it in its actual format. Though, both the format has same speech feature, TDNN ASR system performs better in n-gram match.

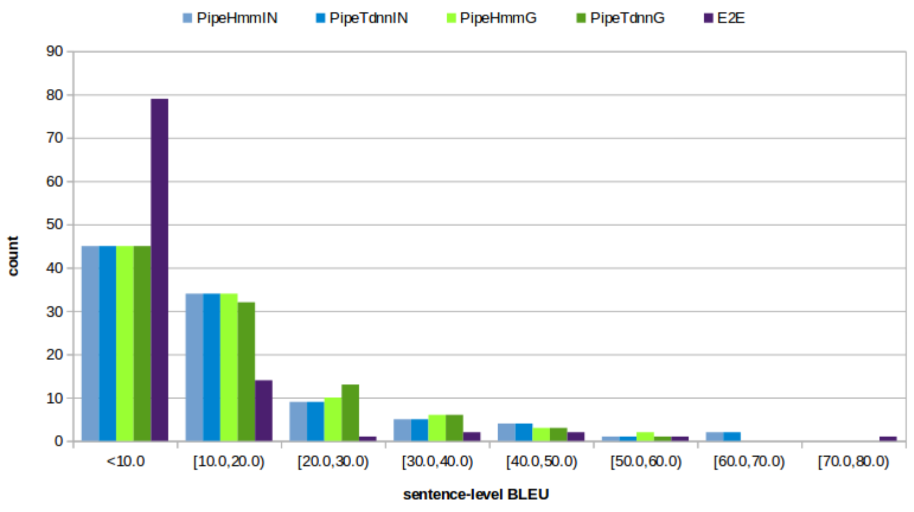


Figure 2: Sentence level BLEU evaluation

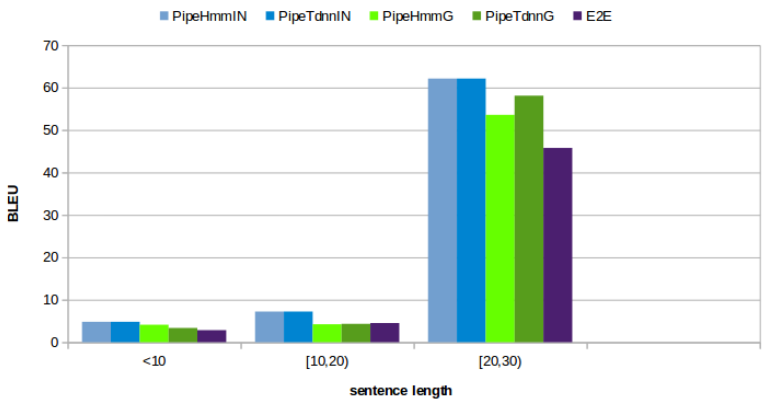


Figure 3: Sentence length BLEU evaluation

The samples **transcript2** to **transcript4** shows some of the examples where ASR systems generates incorrect transcript words (highlighted in “red”) of reference words (highlighted in “green”). From the sample results, it is observed that the ASR systems suffer when the word contains the phoneme “b” (“বন্দ” ~ “bandh”, “ভাবন” ~ “bhavan”, “বাল” ~ “bal”).

A single phoneme in Manipuri could be represented by different graphemes in the Bengali script. One such case is shown in **transcript3** where the ASR systems generate the word “ত্রান্সপোর্ট” (~ “transport”) as “ট্রান্সপোর্ট” (~ “transport”). In **transcript4**, the word “ইমফাল” (~ “imphal”) is a correct representation of the word “ইম্ফাল” (~ “imphal”) where the joined characters are written separately.

As the automatic evaluation metrics are computed at the word level, the cases highlighted in **transcript3** and **transcript4** often led to low evaluation score.

5.3 Sentence Level Evaluation

An analysis of the Manipuri→English S2T translation system is carried out by computing the BLEU score at the sentence level. Figure 2 shows the analysis based on the number of sentences with respect to the BLEU score. While the analysis in Figure 3 shows the performance of the systems with shorter and longer sentences based on the length of the reference sentence.

In Figure 2, the majority of the translations from the E2E model are observed to score a BLEU score of less than 10, while less than half of the translations from the pipeline model scored less than 10. It is interesting to note that the highest sentence level BLEU is achieved by the E2E model even though the overall performance of the pipeline model outperforms the E2E model significantly. A likely cause of the poor performance of end-to-end S2T translation system is the small size of the dataset. The result in Figure 3 shows that the systems perform well with longer sentences [20,30)

source1	আর.তি.আই. এক্টকী মতাংদা খংমিন্নবগী থৌরম পাংথোকখ্রে
reference1	Awareness programme on RTI Act held
GMM-HMM ASR	আর তি আই ঈ কী মতাংদা খংমিন্নবগী থৌরম পাংথোকখ্রে
PipeHmmIN	Awareness programme on Mudra Dayal held held
PipeHmmG	Awareness programme on foot held
TDNN ASR	আর তি আই এক্টকী মতাংদা খংমিন্নবগী থৌরম পাংথোকখ্রে
PipeTdnnIN	Awareness programme on tobacco Dayal control held
PipeTdnnG	Awareness programme on Act held at Moreh
E2E	Awareness programme on RTI Act held at Manipur Press Club , Majorkhul
source2	পাওমীশিংগী মীফম মনুংদা ৱারেপ্পা নাবানা মেডিয়াগী মীওইশিংদা ৱা ঙাংখি
reference2	Wareppa Naba speaks to media persons during press meet
GMM-HMM ASR	পাওমীশিংগী মীফম মনুংদা ৱারেপ্পা নাবানা মেডিয়াগী মীওইশিংদা ৱা ঙাংখি
PipeHmmIN	Ng Ibobi speaks to media persons during press conference
PipeHmmG	Ibobi speaks during media persons during press conference
TDNN ASR	পাওমীশিংগী মীফম মনুংদা ৱারেপ্পা নাবানা মেডিয়াগী মীওইশিংদা ৱা ঙাংখি
PipeTdnnIN	Ng Ibobi speaks to media persons during press conference
PipeTdnnG	Ibobi speaks during media persons during press conference
E2E	Ng . Uttam speaks to media persons during press conference
source3	অপুনবা ইরৈপাক্কি মহৈরোই শিনপাংলুপকী মীহুৎশিংনা মেডিয়াদা ৱা ঙাংলি
reference3	Representatives of Apunba Ireipakki Maheiroi Sinpanglup speaking to the media
GMM-HMM ASR	অপুনবা ইরৈপাক্কি মহৈরোইশিং পান লুপকী মীহুৎশিংনা মেডিয়াদা ৱা ঙাংলি
PipeHmmIN	Representatives of Apunba Ireipakki Maheiroi Sinpanglup speaking to media
PipeHmmG	Representatives of Ukhrul woman speaking during the inaugural ceremony
TDNN ASR	অপুনবা ইরৈপাক্কি মহৈরোই শিনপাংলুপকী মীহুৎশিংনা মেডিয়াদা ৱা ঙাংলি
PipeTdnnIN	Representatives of Apunba Ireipakki Maheiroi Sinpanglup speaking to media
PipeTdnnG	Representatives of Ukhrul woman speaking during the inaugural ceremony
E2E	Representatives of Apunba Ireipakki Maheiroi Sinpanglup speaking to the media

Table 4: Manipuri→English Speech-to-Text translation sample input-output

.

while the sentences with length below 20 score a BLEU score less than 10.

With only very few samples achieving a BLEU score above 50, it is clear that a massive effort is required for the development of Manipuri→English Speech-to-Text translation systems.

5.4 Qualitative Analysis of Manipuri→English S2T Systems

Sample input and output from the pipeline models and E2E Speech-to-Text translation model are shown in Table 4. The grammatical error or incorrect word(s) in the output from our systems are highlighted in "blue".

In the first sample, despite preserving the information moderately, the fluency scale of translation output with the Pipeline-GMM-HMM is worse compared to the other systems. One of the main reason is error propagation from the ASR model where word "এক্টকী" (~ "act-ki") is incorrectly generated as "ঈ কী" (~ "e-ki"). Furthermore, E2E translation model generate additional non-relevant information even though the output sentence is fluent. In the second sample, the named entity word "Wareppa Naba" (a name of a person) is incorrectly generated and is replaced by another name of a person (i.e., Ng Ibobi, Ibobi and Ng . Uttam). In different languages, there are cases where multiple words in one langugae is represented by a single word in another language. One such case is highlighted in the second sample where both the words meet and conference which are synonyms is represented by a single word in Manipuri "মীফম" (~ "mifam"). This often results to low BLEU score as the evaluation metric is computed at the word level n-gram matching and doesn't consider synonyms. The third sample shows the handling of long Manipuri multi-word named entity "অপুনবা ইরৈপাক্কি মহৈরোই শিনপাংলুপকী" (~ "Apunba Ireipakki Maheiroi Sinpanglup-ki"), where the suffix "-ki" is used to denote the possessive noun. It is observed that the multi-word named entity is translated correctly despite the slight variation in the output of

Score	Adequacy	Fluency
1	No information is preserved	Incomprehensible
2	Small amount of information is preserved	Disfluent
3	Moderately preserved information	Non-native
4	All information is preserved	Flawless sentence, and all are correct in terms of grammatical rules

Table 5: Adequacy Fluency scale

	Adequacy	**Fluency**
PipeHmmIN	1.6	2
PipeHmmG	1.5	1.63
PipeTdnnIN	**1.63**	1.98
PipeTdnnG	1.3	1.93
E2E	1.23	**2.83**

Table 6: Human Evaluation of Manipuri$\rightarrow$English S2T Translation Systems.

the ASR system (GMM-HMM). This is the impact of the system trained on in-domain training dataset. However, the translation is not in the line with the NMT system trained on mixed domain training dataset (NMT_g) as the probability distribution got skewed towards the add-on dataset.

5.5 Adequacy and Fluency Analysis of Translation Outputs

Fluency analysis provide evaluation based mainly on grammatical rules. Adequacy indicates information preserved. Adequacy and fluency are measured on a scale of 1 to 4 and the meaning of the various scales are summarized in Table 5. To measure adequacy and fluency, human evaluation on the test dataset from each S2T translation system is carried out. The adequacy and fluency ratings reported by our human evaluators are shown in Table 6.

- Among our translation systems, the pipeline model (PipeTdnnIN) achieves the highest adequacy score. The adequacy score of all the systems are observed to be in correlation with our automatic evaluation.

- The fluency score is observed to be non-correlated with the automatic evaluation scores. In terms of fluency, the end-to-end model achieved the highest score. This indicates that despite not preserving the information of the source language, the system is able to generate a fluent text.

6 Conclusion and Future work

In this work, a comparative study of the conventional pipeline model and end-to-end model of S2T translation on an extremely low-resource Manipuri-English language pair is presented. We also made a comparison of two acoustic models: GMM-HMM and TDNN, for the ASR module. An improvement of 2.53% WER is observed in the ASR model with TDNN compared to GMM-HMM. The TDNN ASR model is observed to be more robust than the GMM-HMM model in terms of n-gram match. The ASR output is fed to a shared NMT system (trained with the in-domain or the additional out of domain dataset) in our pipeline model. In comparison, the translation hypothesis of the pipeline models are comparable in terms of the BLEU score. However, using an NMT system trained with a dataset from mixed domain results to the decrease in the automatic evaluation score. Though the end-to-end S2T translation has various advantages over traditional pipeline models, the limited size of our dataset led to the end-to-end S2T model's low performance compared to the pipeline model. An extensive collection of parallel S2T translation training data is generally required to train such an end-to-end S2T translation model.

In future, we plan to increase the size of the dataset along with the collection of other forms of modalities such as images. We also plan to explore various Speech-to-Text machine translation models to enhance the performance.

References

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

Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. 2018. Low-resource speech-to-text translation. *arXiv preprint arXiv:1803.09164*.

Sameer Bansal, Herman Kamper, Adam Lopez, and Sharon Goldwater. 2017. Towards speech-to-text translation without speech recognition. *arXiv preprint arXiv:1702.03856*.

Alexandre Bérard, Laurent Besacier, Ali Can Kocabiyikoglu, and Olivier Pietquin. 2018. End-to-end automatic speech translation of audiobooks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6224–6228. IEEE.

Alexandre Bérard, Olivier Pietquin, Laurent Besacier, and Christophe Servan. 2016. Listen and Translate: A Proof of Concept for End-to-End Speech-to-Text Translation. In *NIPS Workshop on end-to-end learning for speech and audio processing*, Barcelona, Spain.

Ozan Caglayan, Walid Aransa, Yaxing Wang, Marc Masana, Mercedes García-Martínez, Fethi Bougares, Loïc Barrault, and Joost Van de Weijer. 2016. Does multimodality help human and machine for translation and image captioning? *arXiv preprint arXiv:1605.09186*.

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. In *arXiv preprint arXiv:1406.1078*.

Thangjam Clarinda Devi, Leihaorambam Sarbajit Singh, and Kabita Thaoroijam. 2021. Vowel-based acoustic and prosodic study of three manipuri dialects. In *Advances in Speech and Music Technology*, pages 425–433. Springer.

Long Duong, Antonios Anastasopoulos, David Chiang, Steven Bird, and Trevor Cohn. 2016. An attentional model for speech translation without transcription. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 949–959.

Baban Gain, Dibyanayan Bandyopadhyay, and Asif Ekbal. 2021. Iitp at wat 2021: System description for english-hindi multimodal translation task. In *Proceedings of the 8th Workshop on Asian Translation (WAT2021)*, pages 161–165.

Po-Yao Huang, Frederick Liu, Sz-Rung Shiang, Jean Oh, and Chris Dyer. 2016. Attention-based multimodal neural machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 639–645.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn, Franz J Och, and Daniel Marcu. 2003. Statistical phrase-based translation. Technical report, UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY INFORMATION SCIENCES INST.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Evgeny Matusov, Stephan Kanthak, and Hermann Ney. 2005. On the integration of speech recognition and statistical machine translation. In *Ninth European Conference on Speech Communication and Technology*.

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

Loitongbam Sanayai Meetei, Thoudam Doren Singh, and Sivaji Bandyopadhyay. 2021. Low resource multimodal neural machine translation of english-hindi in news domain. In *Proceedings of the First Workshop on Multimodal Machine Translation for Low Resource Languages (MMTLRL 2021)*, pages 20–29.

Loitongbam Sanayai Meetei, Thoudam Doren Singh, Sivaji Bandyopadhyay, Mihaela Vela, and Josef van Genabith. 2020. English to manipuri and mizo post-editing effort and its impact on low resource machine translation. In *Proceedings of the 17th International Conference on Natural Language Processing (ICON)*, pages 50–59.

Hermann Ney. 1999. Speech translation: Coupling of recognition and translation. In *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258)*, volume 1, pages 517–520. IEEE.

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.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037.

Tanvina Patel, DN Krishna, Noor Fathima, Nisar Shah, C Mahima, Deepak Kumar, and Anuroop Iyengar. 2018. An automatic speech transcription system for manipuri language. In *INTERSPEECH*, pages 2388–2389.

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*.

Matt Post. 2018. A call for clarity in reporting bleu scores. *arXiv preprint arXiv:1804.08771*.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. 2011. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, CONF. IEEE Signal Processing Society.

Laishram Rahul, Loitongbam Sanayai Meetei, and HS Jayanna. 2021. Statistical and neural machine translation for manipuri-english on intelligence domain. In *Advances in Computing and Network Communications*, pages 249–257. Springer.

Laishram Rahul, Salam Nandakishor, L Joyprakash Singh, and SK Dutta. 2013. Design of manipuri keywords spotting system using hmm. In *2013 Fourth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)*, pages 1–3. IEEE.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

Salam Michael Singh, Loitongbam Sanayai Meetei, Thoudam Doren Singh, and Sivaji Bandyopadhyay. 2021. Multiple captions embellished multilingual multi-modal neural machine translation. In *Proceedings of the First Workshop on Multimodal Machine Translation for Low Resource Languages (MMTLRL 2021)*, pages 2–11.

Salam Michael Singh and Thoudam Doren Singh. 2020. Unsupervised neural machine translation for english and manipuri. In *Proceedings of the 3rd Workshop on Technologies for MT of Low Resource Languages*, pages 69–78.

Salam Michael Singh and Thoudam Doren Singh. 2021. Statistical and neural machine translation systems of english to manipuri: A preliminary study. In *Soft Computing and Signal Processing*, pages 203–211. Springer.

Thoudam D Singh and Sivaji Bandyopadhyay. 2010a. Manipuri-english example based machine translation system,". *International Journal of Computational Linguistics and Applications (IJCLA), ISSN*, pages 0976–0962.

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. 2010b. 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.

Andreas Stolcke. 2002. Srilm-an extensible language modeling toolkit. In *Seventh international conference on spoken language processing*.

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.

Changhan Wang, Yun Tang, Xutai Ma, Anne Wu, Dmytro Okhonko, and Juan Pino. 2020. Fairseq s2t: Fast speech-to-text modeling with fairseq. 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: System Demonstrations*, pages 33–39.

Ron J Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen. 2017. Sequence-to-sequence models can directly translate foreign speech. *arXiv preprint arXiv:1703.08581*.

On the Transferability of Massively Multilingual Pretrained Models in the Pretext of the Indo-Aryan and Tibeto-Burman Languages

Salam Michael Singh[1], **Loitongbam Sanayai Meetei**[1], **Alok Singh**[1], **Thoudam Doren Singh**[1], and **Sivaji Bandyopadhyay**[1]

[1]Centre for Natural Language Processing (CNLP) & Dept. of CSE, NIT Silchar, India
{salammichaelcse,loisanayai,alok.rawat478,thoudam.doren,sivaji.cse.ju}@gmail.com

Abstract

In recent times, machine translation models can learn to perform implicit bridging between language pairs never seen explicitly during training and showing that transfer learning helps for languages with constrained resources. This work investigates the low resource machine translation via transfer learning from multilingual pre-trained models i.e. mBART-50 and mT5-base in the pretext of Indo-Aryan (Assamese and Bengali) and Tibeto-Burman (Manipuri) languages via finetuning as a downstream task. Assamese and Manipuri were absent in the pretraining of both mBART-50 and the mT5 models. However, the experimental results attest that the finetuning from these pre-trained models surpasses the multilingual model trained from scratch.

1 Introduction

Recent years have witnessed the growing advances in the field of neural machine translation (NMT) specifically for the resource rich languages. However, NMT requires enormous amount of parallel data in order to have a decent translation system. On the other hand, the low resource languages lacks sufficient amount of parallel data, thus making the translation system far from the production level. Meanwhile, monolingual data is readily available as compared to the parallel data and many works have been done to exploit it, most notably in a semi-supervised approach for data augmentation using self-training (Ueffing, 2006; Zhang and Zong, 2016; He et al., 2020) and back-translation (Sennrich et al., 2013; Edunov et al., 2018). However, these approaches are prone to generate erroneous translations due to the noisy synthetic data and often requires an iterative refinement procedure which is both resource intensive (Hoang et al., 2018) and time consuming process. Unsupervised machine translation (Lample et al., 2018; Artetxe et al., 2018; Lample and Conneau, 2019) on the other hand uses only the monolingual data and do not require any parallel data which appears to be intimidating for a low resource scenario. Additionally, the initial cross-lingual mapping between the two monolingual data requires a maximal amount of vocabulary overlaps which is crucial for a stronger cross-lingual mapping between the source and the target monolingual vector spaces. However, the vocabulary overlaps is maximised only when the two languages are closely related thus making the unsupervised machine translation approach unsuitable for the distant language pairs even if they have large amount of monolingual data (Kim et al., 2020). Moreover, conventional unsupervised systems utilises iterative back-translation for the refinement purpose, thus the unsupervised methods are imposed with the issues of the back-translation (noisy translations and resource intensive). Multilingual neural machine translation (MNMT) (Johnson et al., 2017; Fan et al., 2021) on the other hand supports the translation among multiple languages which has shown to be beneficial for low resource machine translation via the transfer of cross-linguistic information from the higher resource languages (Aharoni et al., 2019; Dabre et al., 2020). This can be facilitated by transferring the trained parameters from a parent model to a child model (Zoph et al., 2016; Nguyen and Chiang, 2017; Kocmi and Bojar, 2018) or through a bridge or pivot language (Dabre et al., 2015; Utiyama and Isahara, 2007; More et al., 2015). However, MNMT can be further simplified by converting it into a single bilingual NMT by jointly training (Firat et al., 2016; Johnson et al., 2017) all the languages. Furthermore, the jointly trained MNMT system is extended with 50 or more languages in a massively multilingual (Aharoni et al., 2019; Fan et al., 2021; Xue et al., 2021) scenario which has shown to improve the low resource machine translation (Dabre et al., 2020) in the presence of the higher resource languages with the advantage of training a single NMT model instead of training separate bilingual models. However, training these massively multilingual models from scratch for every new languages is not feasible both in terms of time and the resource and has negative impact to the environment for training such enormous models which can be coped up via transfer learning where the downstream translation task can be simply finetuned from a large pre-trained model (Liu et al., 2020; Tang et al., 2020; Conneau et al., 2020; Kakwani et al., 2020; Khanuja et al., 2021; Xue et al., 2021; Dabre et al., 2021). Primitive transfer learning in the NLP flourished with the pretrained word embedding vectors (Mikolov et al., 2013; Pennington et al., 2014), followed by the pretrained encoder (Devlin et al., 2019) or decoders or pretraining the full seq2seq model (Liu et al., 2020). These multilingual pretrained models such as the mBART (Liu et al., 2020) and the mT5 (Xue et al., 2021) has shown to benefit the low resource machine translation during the downstream finetuning step. Additionally, these pretrained models can be extended to even new languages (Tang et al., 2020) which was absent during the pretraining process by simply resuming the training with the new language data with the pretrained model checkpoint as a finetuning step and sometimes increasing the BLEU score also.

In our premise, we make use of the mBART-50 (Tang et al., 2020) and the mT5-base (Xue et al., 2021) pretrained models for the English (*en*) to {Assamese (*asm*), Bengali (*bn*) and Manipuri (*mni*)} translation in a one-to-many multilingual setup. All the three languages apart from English are the scheduled languages of India where Assamese and Bengali belong to the Indo-Aryan language family while Manipuri is a Tibeto-Burman language and very few works have been reported in this language most notably (Singh and Bandyopadhyay, 2010; Singh, 2013; Singh and Singh, 2020; Singh et al., 2021; Singh and Singh, 2021; Sanayai Meetei et al., 2020; Rahul et al., 2021; Laitonjam and Ranbir Singh, 2021). Additionally, only the Bengali language is present during the pretraining of both mBART-50 and the mT5-base models while Assamese and Manipuri were absent during the pretraining phase. Hence, the finetuning process involves the transfer learning to totally unseen languages and this work investigates the effect of these pretrained models to the low resource translation task for these unseen languages. We also evaluate our performance on the WAT-2021 MultiIndicMT [1] test set for English to Bengali and Flores-101 test set (Goyal et al., 2021) for the English to (Bengali and Assamese)

2 Multilingual Neural Machine Translation

Multilingual NMT facilitates the translation between multiple languages via pivot based (Dabre et al., 2015), transfer learning (Zoph et al., 2016) or through a jointly trained single NMT model (Johnson et al., 2017). In this work, we utilise the jointly trained single multilingual NMT model. Additionally, this single MNMT can be further divided into three types according to the mapping of the source and the target languages, **Many-to-one** (**m2o**). In this setting, the model is trained to translate multiple source languages into a single target language. **One-to-many** (**o2m**). This MNMT model translates from a single source language to multiple target languages and many-to-many (**m2m**). Here, translation between many source and many target languages is possible. Moreover, as there are several target languages in the **o2m** and **m2m**, a target language tag is typically prepended at the beginning of the source sentence to specify the predicted target language. Given K sentence pairs and L language pairs the training objective of an MNMT model is to maximise the log-likelihood over the whole parallel pairs $\{\mathbf{x}^{(l,k)}, \mathbf{y}^{(l,k)}\}_{k\in(1,\ldots,K_l)}^{l\in(1,\ldots,L)}$ as:

$$\mathcal{L}_\theta = \frac{1}{K}\sum_{l=1}^{L}\sum_{k=1}^{K_l} log\ p(\mathbf{y}^{(l,k)}|\mathbf{x}^{(l,k)};\theta), \qquad (1)$$

where the total parallel sentences $K = \sum_{l=1}^{L} K_l$.

3 Multilingual Pretrained Model

3.1 mBART

The mBART model which follows the sequence-to-sequence (Seq2Seq) pre-training scheme of the BART model and pre-trained on large scale monolingual corpora in 25 languages is used in our work. There are two types of noises used to produce the corrected text by removing the text spans and replacing them with a mask token and secondly by

[1] http://lotus.kuee.kyoto-u.ac.jp/WAT/indic-multilingual

permuting the order of the sentences within each instance. The large-scale pre-training on multiple diverse languages has shown to be helpful at building low-resource NMT systems by being fine-tuned to the target language pair (Dabre et al., 2021; Xue et al., 2021). This also has shown to possess a powerful generalization ability to languages that do not appear in the pre-training corpora.

3.2 mT5

mT5 is a massively multilingual pretrained model variant of Text-to-Text Transfer Transformer (T5) (Raffel et al., 2020). The T5 is trained on a multi-task scenario which is governed by the pre-training on a masked language modeling "span-corruption" objective, in which consecutive input token spans are replaced with a mask token and the model is trained to reconstruct the masked-out tokens.

4 Experimental Setup

4.1 Dataset

The experimentation uses the parallel data from CVIT-PIB (PIB) (Philip et al., 2021) and PMIndia (PMI) (Haddow and Kirefu, 2020) dataset. The Assamese (*asm*) and Manipuri (*mni*) data is curated from PMIndia while Bengali (*bn*) data is taken from both CVIT-PIB and PMIndia dataset. For the development, a small subset of 1000 sentences from the PMI is used for the *mni* and *asm*, while WAT-2021 is used for the *bn* side.

The WAT-2021 test set is in-domain with the PMI and PIB data which are mostly news domain and we also investigate the domain adaptability of these pretrained models on a general domain test set FLORES-101. For this, the *en*-{*asm*, *bn*} translations are finetuned in a multilingual way with the FLORES development data.

4.2 Dataset Preprocessing

The text preprocessing step initially tokenizes the raw texts. English side data is tokenized using the *moses-scripts*[2] while the Indic data are normalized and tokenized using the `IndicNLP` toolkit[3]. Additionally, we do not perform any sort of script conversion for the orthogonality matching as *bn*, *asm* and *mni* all use the same script.

Furthermore, foreign language text are identified and removed using *langid*[4] and their dataset is de-duplicated and ensured that the training data excludes any instances of the development and test sets. Following the work of (Philip et al., 2021), a sentencepiece (Kudo and Richardson, 2018) BPE of 3K subword merges is learnt for each language separately over the normalized and the tokenized text data. However, the vocabulary for *en* is learnt over the combined *en* data. Finally, the union of all the unique tokens is taken to make a common dictionary.

4.3 Training setup

1. **One-to-Many multilingual model trained from scratch (O2M-S)**: A one-to-many multilingual NMT is trained from scratch using transformer with 6 layers of encoders and decoders, 4 attention heads, 512 embedding dimension and a feedforward dimension of 1024. The encoder and decoder are shared and optimised using adam with the betas (0.9, 0.98) with an initial learning rate of 0.0005 which is scheduled using inverse square root with 4000 warmup updates. The training is done using fairseq (Ott et al., 2019) toolkit for 100,000 update steps with a token based batch of batchsize 4000.

2. **mBART+O2M**: We finetune the mBART-50 model in a one-to-many multilingual setup for the *en* to (*asm*, *bn* and *mni*) translation. Furthermore, the fairseq toolkit is used and in particular the multi-simple-epoch task of the fairseq to finetune from mBART-50 pretrained model. The system is an mbart-large architecture and uses the default parameters as in this setup[5] and finetuned for 80,000 update steps.

3. **mT5+O2M**: The mT5-base model is used for the finetuning using the simpletransformers library[6] with the default setup and finetuned for 80,000 update steps.

Furthermore, all the systems are finetuned for another 15,000 update steps upon the FLORES development set after resetting the training optimizers for the domain adaptation as all the systems are

[2]`https://github.com/moses-smt/mosesdecoder/tree/master/scripts`

[3]`https://github.com/anoopkunchukuttan/indic_nlp_library`

[4]`https://github.com/saffsd/langid.py`

[5]`https://github.com/pytorch/fairseq/tree/main/examples/multilingual`

[6]`https://github.com/ThilinaRajapakse/simpletransformers`

trained only on the PMI and PIB data which is a news domain whereas the FLORES-101 test set is a general domain data.

4.4 Comparison with Other Works

This work is compared with the following work evaluated upon the WAT-2021 and FLORES-101 test sets:

1. Ramesh et al. (2021): A multilingual model trained on the largest publicly available parallel corpora.

2. IndicBART (Dabre et al., 2021): A multilingual pretrained model trained on 11 Indic languages trained using mBART objective.

4.5 Evaluation Metrics

1. **Automatic Evaluation**: The automatic evaluation is done using BLEU which is reported over the geometric mean of the 4-gram precision or BLEU-4, ranging from 0-100, with 100 being the highest. The hypothesis for the *en* to {*asm*, *bn*, *mni*} translation evaluation is detokenized and then retokenized using the IndicNLP tokenizer and then evaluated without using any tokenizer in SacreBLEU[7].

2. **Human Evaluation**: Human evaluation is carried out by considering the fluency and adequacy of the translated output. In this pretext, three human translators fluent in English-Manipuri, English-Assamese and English-Bengali are assigned to separately rate each sentence from 1-5 for the fluency and the adequacy criteria. Finally, the sentence wise scores are averaged to get the corpus level score for both the criteria.

5 Experimental Results

Table 1 reports the automatic evaluation scores of the systems based on the BLEU score for the *en* to {*asm*, *bn* and *mni*} one-to-many translations. Both the pretrained models outperforms the multilingual system trained from the scratch (**O2M-S**) across all the translation directions suggesting a successful transfer of information from the pretrained models to the downstream finetuning task.

Additionally, the significant improvement in BLEU score after the finetuning is observed for both the *asm* and *mni* languages which were absent during the pretraining step revealing that these multilingual pretrained models are language independent up to an extent and can be extended to any new languages irrespective of their relatedness from the pretrained languages and thus ideal for a low resource machine translation.

System	asm	bn	mni
O2M-S	11	16.2	19.5
mBART+O2M	15.9	19.8	26.3
mT5+O2M	15.4	18.6	29.2

Table 1: BLEU score evaluated using PMI test set for the en to (asm, bn, mni) translation.

5.1 Comparison With Other Works

Table 2 reports the BLEU score of the trained systems i.e. O2M-S, mBART+O2M and mT5 which is compared with Ramesh et al. (2021) and IndicBART (Dabre et al., 2021) evaluated upon the WAT-2021 and PMI test sets. O2M-S performs the worst amongst all the systems for both the test sets across all the translation directions. For the WAT-2021 test set, mT5+O2M has the best performance followed by Ramesh et al. (2021). Ramesh et al. (2021) is trained using the largest available training data for the Indian languages thus giving an extra edge. On the other hand FLORES test is a general domain data thus making the task more challenging as our systems are trained using only the news domain from PMI and PIB which is reflected in the low BLEU scores of our trained systems for the FLORES test set.

However, IndicBART trained their systems using Samanantar dataset (Ramesh et al., 2021) thus making their system more adaptive to the FLORES domain and surpassing both the mBART+O2M and mT5+O2M models with a

System	Test Set		
	WAT-2021	FLORES	
	bn	asm	bn
Ramesh et al. (2021)	16.0	-	-
IndicBART	11.1	-	30.7
O2M-S	10.7	1.2	2.3
mBART+O2M	14.7	3.5	5.6
mT5+O2M	16.2	2.3	4.8

Table 2: BLEU score of the systems for the en to (asm and bn) evaluated on WAT-2021 and FLORES TEST set.

[7] BLEU+case.mixed+numrefs.1+smooth.exp+tok.none+version.1.5.1

Systems	asm	bn
mBART+O2M w/o FT	2.9	4.6
+5K steps FT	3.1	5.2
+10K steps FT	3.4	5.5
+15K steps FT	3.5	5.6
mT5+O2M w/o FT	0.1	3.3
+5K steps FT	0.3	3.9
+10K steps FT	1.3	4.2
+15K steps FT	1.8	4.8

Table 3: Effect of the BLEU score on the finetuning steps (FT) which is finetuned using FLORES development set for the *en* to (*asm* and *bn*) directions.

whooping 30.7 BLEU score in comparison to the 5.6 and 4.8 BLEU scores for the mBART+O2M and mT5+O2M respectively. Additionally, for the WAT-2021 *en-bn* task, IndicBART performed poorly even though they pretrain an mBART model from the Indic languages and finetune upon it. Furthermore, the low performance of IndicBART on WAT-2021 test reveals two possibilities, i) the finetuning of IndicBART involved more number of languages than our setting, which in turn induced a negative transfer (Dabre et al., 2020) due to the incompatibility of the languages involved thus the degradation in the performance, ii) transfer learning from a massively multilingual pretrained model followed by the multilingual finetuning as in our case is more beneficial than transfer learning from a limited language pretrained model as in the case of IndicBART and we put forward these as a future work.

5.2 Domain Adaptation via Few Shot Learning

The systems in our experimentation are trained on a narrow domain data, thus these systems choke when evaluated on a general domain data. Hence, the systems are further finetuned using the FLORES development set for another 15,000 update steps by resetting the optimisers. The results are reported in Table 3.

It is observed that this domain adaptation using incremental finetuning upon the FLORES development set improves the BLEU score across all the directions for both mBART+O2M and mT5+O2M models. However, this increment is still insignificant in comparison to IndicBART (Dabre et al., 2021) as presented in Table 2.

5.3 Human Evaluation Score

Table 4 reports the human evaluation score of the **O2M-S**, **mBART+O2M** and **mT5+O2M** for the *en* to (*asm*, *bn* and *mni*) translations based on the adequacy and fluency criteria which is evaluated upon the PMI test set. For the *en-mni* translation direction presented in Table 4, the multilingual finetuning over both the pretrained models (**mBART+O2M**) and (**mT5+O2M**) is superior to the multilingual model trained from scratch (**O2M-S**) qualitatively. Additionally, in terms of the adequacy score, **mT5+O2M** performs better than the **mBART+O2M**. However, **mBART+O2M** gives a competitive performance to the **mT5+O2M** in terms of the fluency score.

Moreover, the human evaluation scores correlates well with the automatic scores as reported in Table 1 suggesting the effectiveness of the transfer learning for this unseen language during the pretraining time. On the other hand, **mBART+O2M** has higher human evaluation scores than **mT5+O2M** for the *en-asm* and *en-bn* translations as reported in Table 4. However,

Models	en-mni		en-asm		en-bn	
	Adequacy	Fluency	Adequacy	Fluency	Adequacy	Fluency
O2M-S	3.25	3.07	2.91	3.17	2.75	2.823
mBART+O2M	4.15	4.31	3.82	3.782	3.9122	3.782
mT5+O2M	4.42	4.37	3.801	3.775	3.8622	3.688

Table 4: Human evaluation score evaluated on PMI test set based on the adequacy and fluency criteria.

Source-1	In particular, he mentioned the Buddha and the Ramayana.
Ref	বুদ্ধা অমসুং রামায়নবু মহাক্না অকক্ননা পনখি।
TT	buddhaa amasung raamaayanbu mahakna akaknanaa pankhi.
Gloss	the buddha and the ramayan he particularly mentioned.
O2M-S	মহাক্না পনখি মদুদি মহাক্না বুদ্ধ অমসুং রামাধন বুদ্ধগী মতাংদা পনখি।
TT	mahakna pankhi madudi mahakna buddha amasung raamaadhan buddhagi matanga pankhi.
Gloss	he mentioned that he the buddha and the ramadan buddha's about mentioned.
ET	He mentioned that he mentioned about the Buddha and the Ramadan Buddha.
mBART+O2M	মরুওইনা মহাক্না বুদ্ধ অমসুং রামায়ণগী মতাংদা পনখি।
TT	maruoina mahakna buddha amasung raamaayanagi mataangda pankhi.
Gloss	importantly he the buddha and the ramayan's about mentioned.
ET	Importantly he mentioned about the Buddha and the Ramayan.
mT5+O2M	অকক্নদা মহাক্না বুদ্ধ অমসুং রামায়নগী মতাংদা পনখি।
TT	akaknanaa mahakna buddha amasung raamaayangi mataandga pankhi.
Gloss	in particular he the buddha and the raamaayana about mentioned.
ET	In particular he about mentioned about the buddha and the raamaayana.
Source-2	The Officer Trainees belong to 17 Civil Services, and 3 Services from the Royal Bhutan Civil Service.
Ref	ওফিসর ত্রেনীশিং অদুদা সিবিল সর্বিসকী ১৭ অমসুং রোয়েল ভুতান সিবিল সর্বিসকী অহুম য়াওরি।
TT	ophisar trenishing aduda sibil sarbiski 17 amasung royel bhutan sibil sarbiski ahum yaori.
Gloss	officer trainees in civil services 17 and royal bhutan civil service's three belong to.
O2M-S	ওফিসর ১৭, সিবিল সর্বিসশিং, সিভিল সর্বিসশিং অমসুং রোয়েল সর্বিসশিং অসি ভুতানগী সিভিল সর্বিসশিংদগীনি।
TT	ophisar 17, sibil sarbis-shing, sibhil sarbis-shing amasung royel sarbis-shing asi bhutangi sibhil sarbis-shingdagini.
Gloss	officer 17, civil services, civil services and royal services is bhutan's civil services from.
ET	17 officers, Civil Services, Civil Services and the Royal Services are from Bhutan's Civil Services.
mBART+O2M	ওফিসর ত্রেনীশিং অসি সিবিল সর্বিস ১৭ অমসুং রোয়েল ভুতান সিবিল সর্বিসতগী সর্বিস ৩নি।
TT	ophisar trenishing asi sibil sarbis 17 amasung royel bhutan sibil sarbis-tagi sarbis 3ni.
Gloss	officer trainees these civil service 17 and royal bhutan civil service from service is 3.
ET	These officer trainees are from 17 Civil Services and 3 Services from the Royal Bhutan Civil Service.
mT5+O2M	ওফিসর ত্রেনীশিং অদুদা সিবিল সর্বিসকী ১৭ অমসুং রোয়েল ভুতান সিবিল সর্বিসকী সর্বিস অহুম য়াওরি।
TT	ophisar trenishing aduda sibil sarbiski 17 amasung royel bhutan sibil sarbiski sarbis ahum yaori.
Gloss	officer trainees in civil services 17 and royal bhutan civil service's service three belong to.
ET	The Officer Trainees belong to 17 Civil Services and 3 Services from the Royal Bhutan Civil Service.
Source-3	PMSSY has two components
Ref	পি. এম. এস. এস. ৱাই .গী মশা অনি লৈ
TT	pi. em. ess. ess. yai. gi masa ani lei
Gloss	PMSSY's components two has
O2M-S	PMSSYগী কম্পোনেন্ট অনি লৈ
TT	PMSSYgi kamponeṇt ani lei
Gloss	PMSSY's components two has
ET	PMSSY has two components
mBART+O2M	PMSSYগী কম্পোনেন্ট অনি লৈ
TT	PMSSYgi kamponeṇt ani lei
Gloss	PMSSY's components two has
ET	PMSSY has two components
mT5+O2M	পি এম এস এস এস এস এস হায়বসিগী কম্পোনেন্ট অনি লৈ
TT	pi em ess ess ess ess ess haibasigi kamponent ani lei
Gloss	PMSSSSS so called component two has
ET	The so called PMSSSSS has two components

Table 5: Sample *en-mni* translations by the MT systems

mT5+O2M gives a competitive score in terms of fluency for the *en-asm*. Based on the quantitative and qualitative findings from Table 1 and Table 4 respectively, **mT5+O2M** is beneficial for the *en-mni* translation while for the *en* to (*asm* and *bn*), **mBART+O2M** is found to be effective and we plan to explore these discrepancies in our future work.

6 Qualitative and Error Analysis

6.1 Qualitative Analysis

A qualitative analysis in the form of sample input and output is also presented in Table 5 in addition to the qualitative scores reported in Section 5.3 to compare the translation qualities of the **O2M-S**, **mBART+O2M** and **mT5** for the *en* to *mni* translation of the PMI test set. In doing so, we randomly select three *en* test sentences (Source-1, Source-2 and Source-3) and present the respective translated outputs by the systems. Table 5 contains the following abbreviations: The Roman transliterated *mni* sentence is denoted by TT, Gloss is the *en* word-for-word translation, and the *en* translation for the *mni* sentence is ET.

In the first source sentence (Source-1), **O2M-S** the phrase "*mahakna pankhi*" (he mentioned) twice thus degrading the fluency and the term "*raamaayan* has been wrongly generated as "raamaadhan" (ramadan) which in turn detoriates the adequacy. Similarly, there are several instances where **O2M-S** has generated erroneous words. On the other hand, **mBART+O2M** and **mT5+O2M** made a better translation as compared to the **O2M-S** in terms of both adequacy and fluency. However, **mBART+O2M** translated the source word *In particular* to "*maruoina*" (importantly) while **mT5+O2M** translated into the accurate word "*akaknanaa*" (in particular). Although, the word order has been displaced even after generating the correct word hence the automatic scores which depends upon the exact word overlapping gets penalised. The second (Source-2) and the third source (Source-3) sentences are challenging ones. The Source-2 has complex contextual dependencies which is evident with the struggle to establish the correct dependency relations in the translations of the **O2M-S** and **mBART+O2M** while, **mT5+O2M** is the only system which can successfully establish the meaning of the source sentence along with a fluent translation. Apart from this, the Source-2 contains numerical values *17* and *3* which is successfully translated by all the three systems.

Another challenging instance is the presence of abbreviations in the source sentence and the valid English terms which exists as in the target language. This phenomenon is illustrated in Source-3 translation where all the three systems generated the source word *components* as "*kamponent*" (component) instead of "*masa*" (branch; part; component). Thus, even though the **O2M-S** and **mBART+O2M** generated the correct translation due to token mismatch between the reference and the translations, the BLEU score is penalised. In the same Source-3 sentence, the abbreviation of *PMSSY* is directly copied in the outputs of **O2M-S** and **mBART+O2M** which exists as "*pi. em. ess. ess. yai.*" (PMSSY) in the reference thus degrading the BLEU score. **mT5+O2M** on the other hand generated the extra three extra *S* in the abbreviations and excluded *Y*.

6.2 Error Analysis

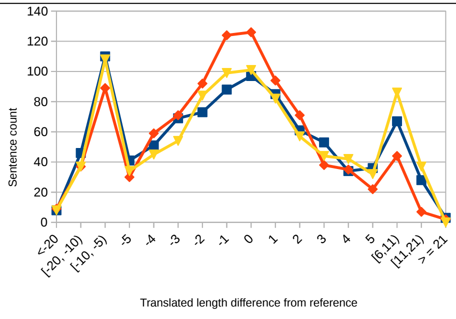
The error analysis of the systems are conducted based on the sentence length. Figure 1A displays the distribution of the difference between the length of the translated output from the reference sentence length of the three systems. Here, the value of "0" at the X-axis signifies that the translated output and the reference sentence are of equal length. In this regard, **mBART+O2M** has the highest count for "0" length difference than both the **mT5+O2M** and **O2M-S** systems across all the translation directions, thus providing the heuristics that the reference and the outputs match word by word which contradicts the superior automatic and human evaluation scores of the **mT5+O2M** than the other two systems for *en* to *mni* translation.

Additionally, for the *en-asm* direction in Figure 1A(i) **O2M-S** and **mT5+O2M** have similar counts for the "0" difference. Furthermore, **mT5+O2M** tends to generate more shorter length sentences than the reference sentence in comparison to the other two systems for all directions, while **O2M-S** generates more longer sentences. Hence, **mBART+O2M** produces more equivalent length to that of the reference than the other two systems.

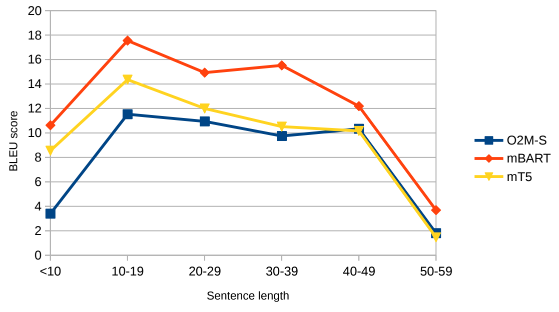
Figure 1B depicts the change in the BLEU score with the varying sentence length. For this, the test sentences are grouped together in buckets based on the sentence length of the reference sentences. For the *en-mni* direction in Figure 1B(iii), **mT5+O2M**

A: Sentence count distribution of hypothesis length difference from reference

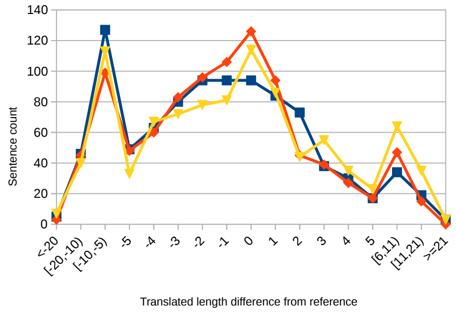
B: BLEU scores of the systems bucketed on test sentence length



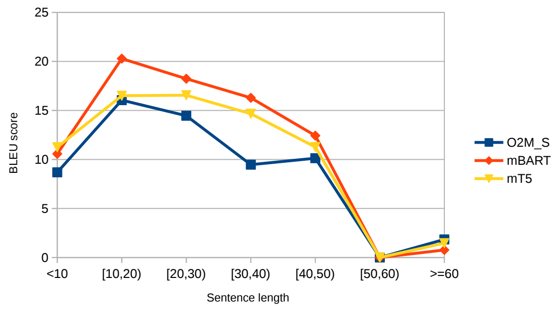
(i) en-asm



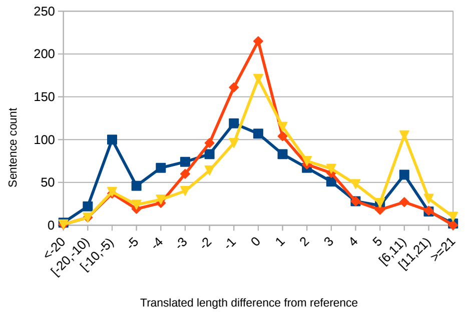
(i) en-asm



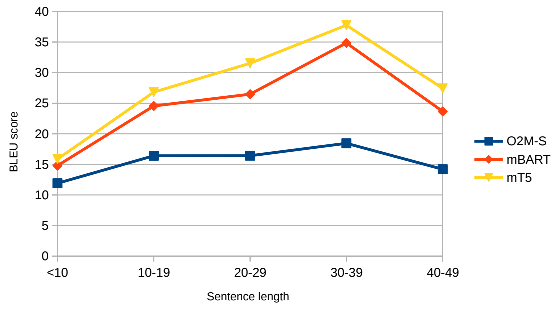
(ii) en-bn



(ii) en-bn



(iii) en-mni



(iii) en-mni

Figure 1: Error analysis of the systems based on the sentence length.

supersedes the other two systems across all the sentence length, followed by the **mBART+O2M**. Meanwhile, **mBART+O2M** is robust to longer sentence length for the *en-asm* (Figure 1B(i)) and similar trend exists in the *en-bn* direction (Figure 1B(ii)) although, **O2M+S** and **mT5+O2M** has higher BLEU scores than **mBART+O2M** for sentences longer than 60 tokens.

7 Conclusion

In this work, we report the findings of the investigation of low resource machine translation via transfer learning from multilingual pretrained models i.e. mBART-50 and mT5-base in the pretext of Indo-Aryan (Assamese and Bengali) and Tibeto-Burman (Manipuri) languages. It is found that the transfer learning from these pretrained multilingual models outperforms the one-to-many model trained from the scratch across all the translation directions in all the test sets thus suggesting the strong transfer of interliguistic information to the downstream finetuning tasks even for the languages absent during the pretraining step. Furthermore, the superiority of finetuning from these pretrained models than the IndicBART for the English to Bengali translation using the WAT-2021 test set suggests that a stronger transfer learning is possible even without linguistic relatedness during the pretraining step or due to the negative transfer of information between the incompatible languages during the multilingual finetuning of IndicBART. Finally, we plan to explore more on the negative transfer and the linguistic relatedness avenue in future focusing on Indian languages.

Acknowledgments

We acknowledge CNLP (Centre for Natural Language Processing) at NIT Silchar for giving access to the lab and the necessary resources. The authors would like to express their gratitude to the anonymous reviewers for their insightful comments. The authors also want to express their gratitude to the volunteers who assisted them with human evaluation tasks.

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*.

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.

Raj Dabre, Chenhui Chu, Fabien Cromieres, Toshiaki Nakazawa, and Sadao Kurohashi. 2015. Large-scale dictionary construction via pivot-based statistical machine translation with significance pruning and neural network features. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, pages 289–297, Shanghai, China.

Raj Dabre, Chenhui Chu, and Anoop Kunchukuttan. 2020. A survey of multilingual neural machine translation. *ACM Comput. Surv.*, 53(5).

Raj Dabre, Himani Shrotriya, Anoop Kunchukuttan, Ratish Puduppully, Mitesh M. Khapra, and Pratyush Kumar. 2021. Indicbart: A pre-trained model for natural language generation of indic languages. *CoRR*, abs/2109.02903.

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.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Michael Auli, and Armand Joulin. 2021. Beyond english-centric multilingual machine translation. *Journal of Machine Learning Research*, 22(107):1–48.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875, San Diego, California. Association for Computational Linguistics.

Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc'Aurelio Ranzato, Francisco Guzmán, and Angela Fan. 2021. The FLORES-101 evaluation benchmark for low-resource and multilingual machine translation. *CoRR*, abs/2106.03193.

Barry Haddow and Faheem Kirefu. 2020. Pmindia - A collection of parallel corpora of languages of india. *CoRR*, abs/2001.09907.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2020. Revisiting self-training for neural sequence generation. In *Proceedings of ICLR*.

Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 18–24, Melbourne, Australia. Association for Computational Linguistics.

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 the Association for Computational Linguistics: EMNLP 2020*, pages 4948–4961, Online. Association for Computational Linguistics.

Simran Khanuja, Diksha Bansal, Sarvesh Mehtani, Savya Khosla, Atreyee Dey, Balaji Gopalan, Dilip Kumar Margam, Pooja Aggarwal, Rajiv Teja Nagipogu, Shachi Dave, Shruti Gupta, Subhash Chandra Bose Gali, Vish Subramanian, and Partha P. Talukdar. 2021. Muril: Multilingual representations for indian languages. *CoRR*, abs/2103.10730.

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.

Tom Kocmi and Ondřej Bojar. 2018. Trivial transfer learning for low-resource neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 244–252, Belgium, Brussels. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Lenin Laitonjam and Sanasam Ranbir Singh. 2021. Manipuri-English machine translation using comparable corpus. In *Proceedings of the 4th Workshop on Technologies for MT of Low Resource Languages (LoResMT2021)*, pages 78–88, Virtual. Association for Machine Translation in the Americas.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *Advances in Neural Information Processing Systems (NeurIPS)*.

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)*.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

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*, volume 26. Curran Associates, Inc.

Rohit More, Anoop Kunchukuttan, Pushpak Bhattacharyya, and Raj Dabre. 2015. Augmenting pivot based SMT with word segmentation. In *Proceedings of the 12th International Conference on Natural Language Processing*, pages 303–307, Trivandrum, India. NLP Association of India.

Toan Q. Nguyen and David Chiang. 2017. Transfer learning across low-resource, related languages for neural machine translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 296–301, Taipei, Taiwan. Asian Federation of Natural Language Processing.

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 the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

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.

Jerin Philip, Shashank Siripragada, Vinay P. Namboodiri, and C. V. Jawahar. 2021. Revisiting low resource status of indian languages in machine translation. In *8th ACM IKDD CODS and 26th COMAD*, CODS COMAD 2021, page 178–187, New York, NY, USA. Association for Computing Machinery.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Laishram Rahul, Loitongbam Sanayai Meetei, and H. S. Jayanna. 2021. Statistical and neural machine translation for manipuri-english on intelligence domain. In *Advances in Computing and Network Communications*, pages 249–257, Singapore. Springer Singapore.

Gowtham Ramesh, Sumanth Doddapaneni, Aravinth Bheemaraj, Mayank Jobanputra, Raghavan AK, Ajitesh Sharma, Sujit Sahoo, Harshita Diddee, Mahalakshmi J, Divyanshu Kakwani, Navneet Kumar, Aswin Pradeep, Kumar Deepak, Vivek Raghavan, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh Shantadevi Khapra. 2021. Samanantar: The largest publicly available parallel corpora collection for 11 indic languages. *CoRR*, abs/2104.05596.

Loitongbam Sanayai Meetei, Thoudam Doren Singh, Sivaji Bandyopadhyay, Mihaela Vela, and Josef van Genabith. 2020. English to Manipuri and mizo post-editing effort and its impact on low resource machine translation. In *Proceedings of the 17th International Conference on Natural Language Processing (ICON)*, pages 50–59, Indian Institute of Technology Patna, Patna, India. NLP Association of India (NLPAI).

Rico Sennrich, Martin Volk, and Gerold Schneider. 2013. Exploiting synergies between open resources for German dependency parsing, POS-tagging, and morphological analysis. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, pages 601–609, Hissar, Bulgaria. INCOMA Ltd. Shoumen, BULGARIA.

Salam Michael Singh, Loitongbam Sanayai Meetei, Thoudam Doren Singh, and Sivaji Bandyopadhyay. 2021. Multiple captions embellished multilingual multi-modal neural machine translation. In *Proceedings of the First Workshop on Multimodal Machine Translation for Low Resource Languages (MMTLRL 2021)*, pages 2–11, Online (Virtual Mode). INCOMA Ltd.

Salam Michael Singh and Thoudam Doren Singh. 2020. Unsupervised neural machine translation for English and Manipuri. In *Proceedings of the 3rd Workshop on Technologies for MT of Low Resource Languages*, pages 69–78, Suzhou, China. Association for Computational Linguistics.

Salam Michael Singh and Thoudam Doren Singh. 2021. Statistical and neural machine translation systems of english to manipuri: A preliminary study. In *Soft Computing and Signal Processing*, pages 203–211, Singapore. Springer Singapore.

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, Atlanta, Georgia. Association for Computational Linguistics.

Thoudam Doren Singh and Sivaji Bandyopadhyay. 2010. 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, Beijing, China. Coling 2010 Organizing Committee.

Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2020. Multilingual translation with extensible multilingual pretraining and finetuning. *CoRR*, abs/2008.00401.

Nicola Ueffing. 2006. Using monolingual source-language data to improve MT performance. In *Proceedings of the Third International Workshop on Spoken Language Translation: Papers*, Kyoto, Japan.

Masao Utiyama and Hitoshi Isahara. 2007. A comparison of pivot methods for phrase-based statistical machine translation. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 484–491, Rochester, New York. Association for Computational Linguistics.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, Austin, Texas. Association for Computational Linguistics.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas. Association for Computational Linguistics.

Generating Slogans with Linguistic Features using Sequence-to-Sequence Transformer

Yeoun Yi
Seoul National University
yeounyi@gmail.com

Hyopil Shin
Seoul National University
hpshin@snu.ac.kr

Abstract

Previous work generating slogans depended on templates or summaries of company descriptions, making it difficult to generate slogans with linguistic features. We present LexPOS, a sequence-to-sequence transformer model that generates slogans given phonetic and structural information. Our model searches for phonetically similar words given user keywords. Both the sound-alike words and user keywords become lexical constraints for generation. For structural repetition, we use POS constraints. Users can specify any repeated phrase structure by POS tags. Our model-generated slogans are more relevant to the original slogans than those of baseline models. They also show phonetic and structural repetition during inference, representative features of memorable slogans.

1 Introduction

Advertising slogans share many linguistic features, such as phonetic or structural repetition (Musté et al. (2015)). These factors make slogans more memorable (Reece et al. (1994)). However, most previous works on slogan generation depended on templates or summaries of company descriptions, making it difficult to generate slogans with linguistic features.

We present LexPOS, a sequence-to-sequence (seq2seq) transformer model with an additional POS encoder. It models the phonetic and structural repetition in slogans, using lexical and POS constraints. When given keywords and POS tags of the desired output structure as input, the model finds words that sound and mean similar to the user keywords. The model-generated slogans include both the user keywords and one sound-alike word. They also reflect the POS constraints. For instance, if a user inputs the word 'cake' and ['VERB', 'DET', 'NOUN', 'PUNCT', 'VERB', 'DET', 'NOUN', 'PUNCT'], the output could be 'Bake a cake, bake a smile'. It includes the word 'cake' and its sound-alike word 'bake' and has repeated verb phrases. The source code, pretrained weights, and data are available online[1].

This paper primarily makes the following contributions:

- Generating slogans taking linguistic features into account.
- Utilizing a pretraining method of BART and T5 to model lexical constraints.
- Proposing a novel approach to model structural constraints by adding a POS encoder.

2 Previous Work

Most of the previous work in slogan generation focused on modifying templates. BRAINSUP, proposed by Özbal et al. (2014), is the first study to generate customized slogans with lexical, emotional, and domain constraints. BRAINSUP utilizes morpho-syntactic patterns mined from corpus as templates. It first selects the most compatible template and fills the empty slots in the template according to user specifications. Before returning the results, it evaluates the candidate slogans with various metrics, including phonetic repetition. However, it can only determine whether the same phonetic features were used or not.

[1] https://github.com/yeounyi/LexPOS

Munigala et al. (2018) presented a model to generate persuasive sentences from fashion product descriptions. It expands fashion-related keywords from inputs and generates sentences using a domain-specific neural language model (LM). Keywords from inputs, expanded keywords, and common functional words are the only candidates at each time step of the LM. The overall perplexity is minimized with beam search. The limitation is that only imperatives can be generated, as the sentences always begin with a verb.

Jin et al. (2021) introduced a sequence-to-sequence transformer model to generate diverse slogans from company descriptions. They considered slogan generation as abstractive summarization of company descriptions and chose the BART-style encoder-decoder model (Lewis et al. (2020)) with a bidirectional encoder and an autoregressive decoder. To prevent unrelated company names from appearing in slogans, they delexicalized all the company names. In addition, they trained a model conditioned on the first words' POS tag, generating syntactically diverse slogans.

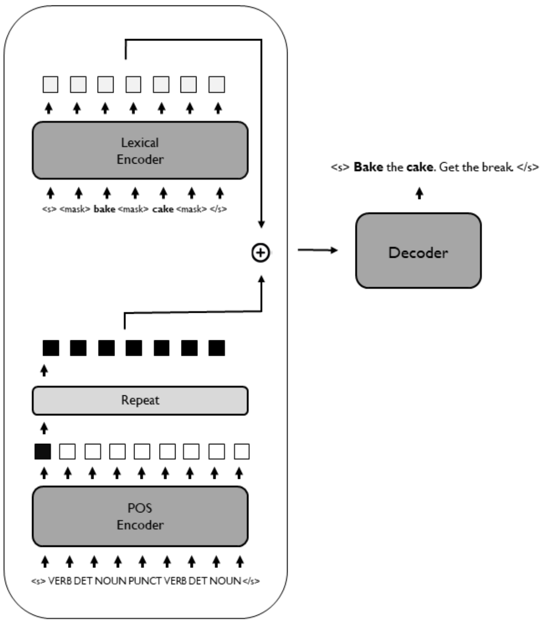
Unlike previous works, we do focus on linguistic features and not depend on templates at the same time. We take phonetic and structural repetition into account, factors that make slogans memorable and unique.

3 Model

Our model first forms the lexical constraints. During training, it uses the given lexical constraints as it is. During inference, it searches for sound-alike words of user keywords. We use the phonetic vector representation proposed by Parrish (2017). The phonetic vector uses interleaved phonetic feature bigrams extracted from phonetic transcriptions and it covers all the words in CMU Pronouncing Dictionary [2] . The model also considers the semantic similarity of sound-alike words, to improve the naturalness of the outputs. We use pretrained Glove embeddings (Pennington et al. 2014) for semantic similarity. After we compute cosine similarity to select the top 100 phonetically similar words, we sort them in semantic similarity. We exclude words that are not present in Glove embeddings (Pennington et al. 2014) or BART tokenizer vocabulary, not to use unfamiliar words. We select the first three words to each form the lexical constraints, together with user keywords. Unlike lexical constraints, POS constraints don't need further processing during training and inference. POS constraints during inference can be manually specified or popular POS structures from data would be recommended.

After processing the lexical constraints, the Transformer architecture (Vaswani et al., 2017) comes in. The Transformer architecture has achieved state-of-the-art results on various natural language processing tasks. We apply a Transformer-based sequence-to-sequence model because we need encoders for constraints and decoders for generation. To leverage the power of pretrained transformers, we utilized the pretrained weights of BART and T5 (Raffel et al., 2020) released by HuggingFace[3]. We choose BART and T5 because both models were pretrained by denoising consecutive spans of corrupted tokens, meaning they can generate natural sentences using lexical constraints.

The only architectural difference is that our model has an additional encoder. One encoder encodes the lexical constraints, and the other encodes the POS constraints. The weights of the POS encoder are randomly initialized, and the vocabulary size of the POS encoder is limited to 20. The vocabulary includes the spaCy[4] POS tags and `<s>`, `</s>`, `<pad>` tokens.


Lexical Encoder
<s> <mask> bake <mask> cake <mask> </s>
<s> Bake the cake. Get the break </s>
Decoder
Repeat
POS Encoder
<s> VERB DET NOUN PUNCT VERB DET NOUN </s>

[2] http://www.speech.cs.cmu.edu/cgi-bin/cmudict

[3] https://huggingface.co/transformers

[4] https://spacy.io/

Slogan	Lexical Input	POS Input
Breakfast of Champions.	<mask> **breakfast** <mask> **champions** <mask>	['NOUN', 'ADP', 'NOUN', 'PUNCT']
The Best a Man Can Get.	<mask> **best** <mask> **man** <mask>	['DET', 'ADJ', 'DET', 'NOUN', 'AUX', 'VERB', 'PUNCT']
Think Different.	<mask> **different** <mask> **think** <mask>	['VERB', 'ADJ', 'PUNCT']
America Runs on <name>.	<mask> **<name>** <mask> **america** <mask> **runs** <mask>	['PROPN', 'VERB', 'ADP', 'PROPN', 'PUNCT']

Table 1: Example of data. Lexical constraints are bolded in lexical inputs. Special tokens are omitted.

Figure 1: Architecture of LexPOS model.

To incorporate the POS constraints into the rest of the model, the last hidden states of <s> token in the POS encoder are repeated with the length of the last hidden states in the lexical encoder. We choose the last hidden state of <s> token because it is widely assumed to include representative information of all tokens. These two hidden states are summed and given to the decoder. Then, the decoder generates slogans with the given lexical and POS constraints. Figure 1 presents the architecture of our proposed model.

4 Data

Our training objective is to implement lexical and POS constraints. The desired model-generated slogans should follow the lexical constraints and the POS structural constraints.

We crawl 30,759 unique slogans from online slogan databases such as Textart.ru[5], Slogans Hub[6], Slogan List[7], Think Slogans[8], and Slogans Point[9]. Unlike previous works focusing on commercial slogans, our dataset covers both commercial and public slogans. Public slogans include slogans for health, women's rights, the environment, and more. 45.43% of our slogans are commercial, 54.56% are public. Company names in commercial slogans are delexicalized using a custom special token <name>, following Jin et al. (2021). We reserve 20% of the data for validation.

The lexical inputs are lexical constraints surrounded by <mask> tokens. Just like the pretraining method of BART and T5, our model predicts consecutive spans of <mask> tokens.

The lexical constraints are limited to verbs, nouns, proper nouns, and adjectives. We extract them from original slogans using spaCy. Then, we randomly delete lexical constraints, when there are 5 or more of them to keep the average ratio of the number of lexical constraints to the total number of words in the original slogan below 50%. The average ratio is 41.90%.

Unlike the pretraining method of BART and T5, we shuffle the lexical constraints to make the model predict natural ordering. If we don't shuffle them, we need to permutate lexical constraints during inference. For instance, if the user keyword is 'cake' and its sound-alike word is 'bake', we need both '<mask> cake <mask> bake <mask>' and '<mask> bake <mask> cake <mask>' as lexical inputs. The number of permutations would increase dramatically as the number of user keywords increase. To address this issue, we shuffle the lexical constraints.

The POS inputs are POS constraints themselves. We use spaCy POS tagging results of the original slogan as POS inputs. Table 1 shows the example of data.

5 Experiments

Following previous work, we conduct a quantitative evaluation using ROUGE (Lin (2004)) F1 scores and compare our model with the original sequence-to-sequence model baselines. We also compute the included lexical constraints rates and POS F1 scores to check how well the given constraints are applied. The included lexical constraints rates are the rates of the lexical constraints included in model-generated slogans.

[5] http://www.textart.ru/database/slogan/map.html
[6] https://sloganshub.org/
[7] https://www.sloganlist.com/
[8] https://www.thinkslogans.com/
[9] http://www.sloganspoint.com/

	Rouge 1	Rouge 2	Rouge L	Included Lexical Constraints Rates	POS F1
baseline BART	0.4894	0.2282	0.4577	**0.9515**	0.7992
baseline T5	0.4461	0.1805	0.4150	0.9406	0.7737
LexPOS BART	**0.6204**	**0.3703**	**0.5921**	0.9469	**0.9176**
LexPOS T5	0.5700	0.3039	0.5339	0.9399	0.8809

Table 2: The quantitative evaluation of various models. Best scores are bolded.

POS F1 scores are computed by comparing the POS inputs and POS tagging results of model-generated slogans.

Table 2 presents the quantitative evaluation result. Our best model achieved a ROUGE-1/-2/-L F1 score of 62.04/37.03/59.21, 94.69 for the included lexical constraints rates, and 91.76 for POS F1 scores. The performance discrepancy between BART and T5 could be explained by their pretraining methods. BART was pretrained using Sentence Permutation, which restores the original order of shuffled sentences, while T5 was not.

Table 3 shows the sample of generated slogans from validation data. The results of the LexPOS model are more relevant to the original slogans.

Gold: The Power of being Global. **BART:** Global Power. **LexPOS BART:** The power of global.
Gold: <name>. Keep Walking. **BART:** I'm walking <name>. **LexPOS BART:** <name>. Walking on.
Gold: How about a nice <name>? **BART:** Be nice to <name>. **LexPOS BART:** Always be a nice <name>.
Gold: All things are difficult before they are easy. **BART:** Difficult things are never easy. **LexPOS BART:** The difficult things are made easy by you.
Gold: Some bruises are on the inside. Stop bullying. **BART:** Stop bullying on the inside and stop bruises on the outside. **LexPOS BART:** The bruises are on the inside, stop bullying.

Table 3: Sample generated slogans from validation data. "Gold" is the original slogan.

Table 4 shows the inference results. We use beam search and adjust the temperature to generate natural slogans.

Keywords: bakery, sandwich **POS:** [VERB, DET, NOUN, PUNCT, VERB, DET, NOUN, PUNCT] **Output: Switch** the bakery, the **sandwich switch**es.
Keywords: airline, cheap **POS:** [ADJ, NOUN, PUNCT, ADJ, NOUN, PUNCT] **Output:** Fast Airline. **Keep Cheap.**
Keywords: save, energy **POS:** [VERB, DET, NOUN, PUNCT, VERB, DET, NOUN, PUNCT] **Output: Save** energy, **save face**s.
Keywords: brunch, cafe **POS:** [NOUN, ADP, NOUN, PUNCT, NOUN, ADP, NOUN, PUNCT] **Output: Brunch** at **Brightness**, Crunch at Cafe.
Keywords: unique, fashion, brand **POS:** [NOUN, ADP, NOUN, PUNCT, NOUN, ADP, NOUN, PUNCT] **Output:** Brand of unique **fashion. Passion** for **fashion.**

Table 4: Sample generated slogans from user keywords and POS constraints. One of the user keywords and its sound-alike word are bolded.

The model-generated slogans include both the user keywords and one selected sound-alike word, fully reflecting users' intentions. The results also show phonetic and structural repetition, representative features of memorable slogans.

6 Conclusion

In this work, we generate slogans with phonetic and structural repetition using LexPOS model, a transformer-based sequence-to-sequence model with an additional POS encoder. It generates slogans using sound-alike words given user keywords. The model-generated slogans also follow structural constraints thanks to the POS encoder. To our knowledge, it is the first model to generate slogans without templates, taking linguistic features into account. Future work should implement other linguistic features shown in slogans.

References

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems, pp. 6000-6010.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, M. Matena, Yanqi Zhou, W. Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res., 21:140:1-140:67.

Jin, Y., Bhatia, A., Wanvarie, D., and Le, P. T. V. 2021. Toward Improving Coherence and Diversity of Slogan Generation. arXiv.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 7871-7880.

Lin, Chin-Yew. 2004. ROUGE: A Package for Automatic Evaluation of summaries. Proceedings of the ACL Workshop: Text Summarization Braches Out 2004. 10.

Munigala, V., Mishra, A., Tamilselvam, S. G., Khare, S., Dasgupta, R., and Sankaran, A. 2018. PersuAIDE! An Adaptive Persuasive Text Generation System for Fashion Domain. In Companion Proceedings of the Web Conference 2018, pp. 335-342, Lyon, France.

Musté, P., Stuart, K., & Botella, A. 2015. Linguistic Choice in a Corpus of Brand Slogans: Repetition or Variation. Procedia - Social and Behavioral Sciences, 198, 350-358.

Özbal, G., D., and Strapparava, C. 2013. BRAINSUP: Brainstorming Support for Creative Sentence Generation. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1446-1455, Sofia, Bulgaria.

Parrish, A. 2017. Poetic Sound Similarity Vectors Using Phonetic Features. In The AIIDE-17 Workshop on Experimental AI in Games, pp. 99-106.

Pennington, Jeffrey & Socher, Richard & Manning, Christopher. 2014. Glove: Global Vectors for Word Representation. EMNLP. 14. 1532-1543. 10.3115/v1/D14-1162.

Reece, B.B., B.G. Vanden Bergh, and H. Li. 1994. What makes a slogan memorable and who remembers it? Journal of Current Issues and Research in Advertising 16, no. 2: 41–57.

Using Integrated Gradients and Constituency Parse Trees to explain Linguistic Acceptability learnt by BERT

Anmol Nayak, Hari Prasad Timmapathini
ARiSE Labs at Bosch
{Anmol.Nayak, HariPrasad.Timmapathini}@in.bosch.com

Abstract

Linguistic Acceptability is the task of determining whether a sentence is grammatical or ungrammatical. It has applications in several use cases like Question-Answering, Natural Language Generation, Neural Machine Translation, where grammatical correctness is crucial. In this paper we aim to understand the decision-making process of BERT (Devlin et al., 2019) in distinguishing between Linguistically Acceptable sentences (LA) and Linguistically Unacceptable sentences (LUA). We leverage Layer Integrated Gradients Attribution Scores (LIG) to explain the Linguistic Acceptability criteria that are learnt by BERT on the Corpus of Linguistic Acceptability (CoLA) (Warstadt et al., 2018) benchmark dataset. Our experiments on 5 categories of sentences lead to the following interesting findings: 1) LIG for LA are significantly smaller in comparison to LUA, 2) There are specific subtrees of the Constituency Parse Tree (CPT) for LA and LUA which contribute larger LIG, 3) Across the different categories of sentences we observed around 88% to 100% of the Correctly classified sentences had positive LIG, indicating a strong positive relationship to the prediction confidence of the model, and 4) Around 43% of the Misclassified sentences had negative LIG, which we believe can become correctly classified sentences if the LIG are parameterized in the loss function of the model.

1 Introduction

Linguistic acceptability is an important criteria in Natural Language Processing and is one of the tasks in the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018). With the evolution of language encoders like BERT (which leverages the multi-head self-attention mechanism (Vaswani et al., 2017) in its architecture) that have been a breakthrough in language understanding and achieved state-of-the-art results, the field of probing these architectures for understanding their behaviours has become important.

While there have been several works on interpreting and understanding the different layers of BERT with respect to lexical, syntactic and semantic behaviours (Jawahar et al., 2019; Lin et al., 2019; Clark et al., 2019; Vashishth et al., 2019; Rogers et al., 2020), the focus on explaining the linguistic acceptability (grammaticality) learnt by BERT has been sparse. Some of the recent works have used probing tasks to understand the model's knowledge on particular grammatical features (Shi et al., 2016; Ettinger et al., 2016; Tenney et al., 2019), relying on language model probabilities to judge grammatical acceptability on sentences that differ minimally (Marvin and Linzen, 2018; Wilcox et al., 2019), or probing the model's by training with boolean grammaticality judgement objectives (Linzen et al., 2016; Warstadt et al., 2018; Kann et al., 2019; Warstadt et al., 2019). These methods have made significant progress in uncovering that BERT has indeed learnt various aspects of grammatical knowledge, however their focus has not been on explaining the black box details of how BERT arrives at a grammaticality judgement. Our paper attempts to address this by explaining the model's linguistic acceptability judgement with LIG and CPT (a type of grammar tree) representations.

Attention mechanism based methods (Bahdanau et al., 2014; Vaswani et al., 2017) provide interpretable understanding of the model's behaviour, however the attention scores cannot be solely relied upon since a feature could influence the output in multiple ways (for e.g. through memory cells, recurrent states etc. in LSTM networks). Feature attribution methods aim to understand the relationship between the model's output and the input features. They are helpful in interpreting the black-

Sentence	Category
rebecca saw the play .	CIA^{LA}
the play saw .	CIA^{LUA}
i surprised myself .	RAA^{LA}
i surprised himself .	RAA^{LUA}
the boy is here .	SVA^{LA}
the boy are here .	SVA^{LUA}
michael read the book .	SVO^{LA}
michael the book read .	SVO^{LUA}
what did rebecca read ?	WHE^{LA}
what did rebecca read the book ?	WHE^{LUA}

Table 1: Sample sentences across the 5 categories from the CoLA Targeted Test Sets.

box details of neural networks and provide insights that can be used to improve model performance. While previous feature attribution methods such as DeepLift (Shrikumar et al., 2016, 2017), Layer-wise relevance propagation (Binder et al., 2016) and LIME (Ribeiro et al., 2016) have provided interesting frameworks, they break at least one of the two axioms that are fundamental for attribution methods, namely *Sensitivity* and *Implementation Invariance* (Sundararajan et al., 2017).

In our paper we have chosen the Integrated Gradients (IG) (Sundararajan et al., 2017) attribution method as it satisfies both the aforementioned axioms. IG is a post-hoc interpretability technique which aggregates the gradients of the input by interpolating in small steps along the straight line between a baseline (typically a vector with all zeros) and the input. A large positive or negative IG score indicates that the feature strongly increases or decreases the network output respectively, while a score close to zero indicates that the feature does not influence the network output. This can also be understood as follows: a positive score indicates that the feature tends to agree with the model's prediction, while a negative score indicates that the feature tends to disagree with the model's prediction. LIG are computed as the IG between the model output and a particular layer's input or output. Our work attempts to answer the following Research Questions:

1. Can LIG of a Constituency Parse Tree (CPT) give insights on LA vs LUA?
2. Can LIG be reliably used to explain the Linguistic Acceptability criteria learnt by BERT?
3. Is there a relationship between LIG and the prediction confidence of the model?

2 Experiment Setup

CoLA dataset sentences have a boolean acceptability judgement, namely LA and LUA. We have used the fine-tuned BERT-Base-Uncased-CoLA model (12 encoder layers with 12 attention heads) provided by TextAttack (Morris et al., 2020), the Captum PyTorch Interpretability library (Kokhlikyan et al., 2020) for computing LIG and the Stanford CoreNLP toolkit (version 4.2.1) (Manning et al., 2014) for constructing the CPT. Integrated Gradients (IG) across the i^{th} dimension of input x and baseline x' are computed as follows:

$$IG = (x_i - x'_i) \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} d\alpha$$

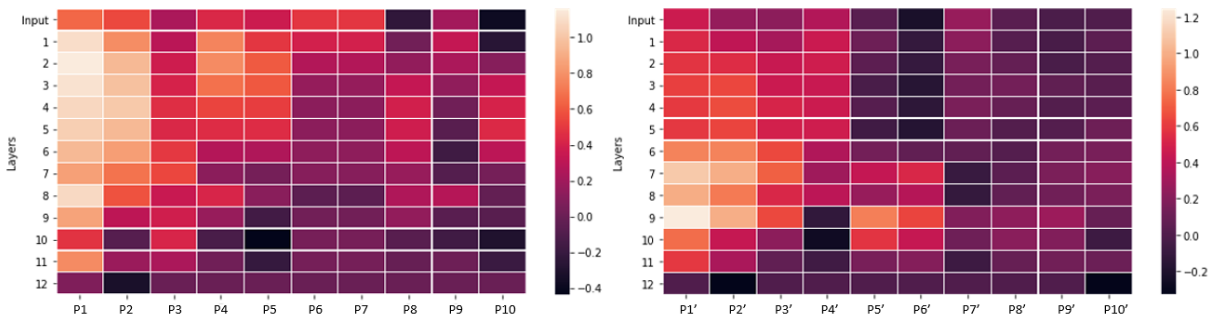
Captum library approximates the above integral using the Gauss-Legendre approximation algorithm over 50 uniform steps of $\alpha \in [0, 1]$. The baseline was selected as a 768 dimension zero vector. The attribution score for each word is summed across the dimensions (768 in the case of BERT-Base) and normalized using the Euclidean norm of the scores of all the words in the sentence.

We analyzed 5 different categories of sentences within the CoLA Targeted Test Sets: Causative-Inchoative Alternation (CIA), Reflexive-Antecedent Agreement (RAA), Subject-Verb Agreement (SVA), Subject-Verb-Object (SVO) and Wh-Extraction (WHE). A few sample sentences across the categories can be seen in Table 1.

The primary focus of our experiments relied on the LIG computed between the predicted class logit and the token embedding of the words. Further we also computed LIG heatmaps with respect to the Input (Token + Segment + Position) embedding and across the 12 Encoder layer embeddings of BERT to analyze the LIG characteristics. Figure 1 shows the LIG heatmaps of the top 10 CPT patterns for the LA and LUA in the WHE category. The unique CPT patterns were extracted for the correctly classified sentences of each category, corresponding to which the LIG of each subtree were computed. LIG of a subtree is equal to the sum of the LIG of the words appearing as leaf nodes in the subtree. The results in Table 2, Table 3, Figure 2 and Figure 3 represent the LIG computed between the predicted class logit and the token embedding of the words. For Out-Of-Vocabulary words (OOV), the LIG are summed across its tokenized sub-words.

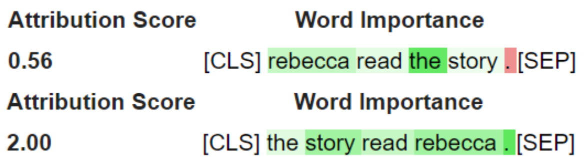
CPT Pattern	Avg. LIG	Category
(S(NP(NN))(VP(VBD)**(NP(DT)(NN))**)(.))	0.065	CIA^{LA}
(S(NP(DT)(NN))(VP(VBD))(.))	1.351	CIA^{LUA}
(S(NP(PRP))(VP(VBD)**(NP(PRP))**)(.))	0.061	RAA^{LA}
(S(NP(PRP))(VP(VBD)(NP(PRP)))(.))	0.926	RAA^{LUA}
(S**(NP(DT)(NN))**(VP(VBZ)(ADVP(RB)))(.))	0.064	SVA^{LA}
(S(NP(DT)**(NN)**)(VP(VBP)(ADJP(JJ)))(.))	1.067	SVA^{LUA}
(S(NP(NN))(VP(VBD)**(NP(DT)(NN))**)(.))	-0.012	SVO^{LA}
(S(NP(NP(NN))(NP(DT)(NN)))(VP**(VBD)**)(.))	1.226	SVO^{LUA}
(SBARQ(WHNP(WP))(SQ(VBD)(NP(NN))**(VP(VB)**))(.))	0.205	WHE^{LA}
(SBARQ(WHNP(WP))(SQ(VBD)(NP(NN))(VP(VB)(NP(DT)(NN))))(.))	1.394	WHE^{LUA}

Table 2: Average normalized LIG of most frequent CPT patterns on 5 categories of Correctly classified CoLA Targeted Test Sets sentences. Subtrees in bold have the largest LIG in the respective categories.



P1: (SBARQ(WHNP(WP))(SQ(VBD)(NP(NNS))(VP(VB)))(.)) ; P2: (SBARQ(WHNP(WP))(SQ(VBD)(NP(NN))(VP(VB)))(.)) ; P3: (.) ; P4: (SQ(VBD)(NP(NNS))(VP(VB))) ;
P5: (SQ(VBD)(NP(NN))(VP(VB))) ; P6: (VB) ; P7: (VP(VB)) ; P8: (NP(NNS)) ; P9: (VBD) ; P10: (NN) ; P1′: (SBARQ(WHNP(WP))(SQ(VBD)(NP(NNS))(VP(VB)(NP(DT)(NN))))(.));
P2′: (SBARQ(WHNP(WP))(SQ(VBD)(NP(NN))(VP(VB)(NP(DT)(NN))))(.)) P3′: (SQ(VBD)(NP(NN))(VP(VB)(NP(DT)(NN)))) ; P4′: (VBD) ; P5′: (VP(VB)(NP(DT)(NN))) ;
P6′: (NP(DT)(NN)) ; P7′: (VB) ; P8′: (.) ; P9′: (NN) ; P10′: (WP)

Figure 1: LIG heatmaps of the top 10 scoring CPT patterns ranked in descending order based on averaged LIG across the different BERT layers for LA (Left) and LUA (Right) of the WHE category.



Attribution Score	Word Importance
0.56	[CLS] rebecca read the story . [SEP]

Attribution Score	Word Importance
2.00	[CLS] the story read rebecca . [SEP]

Figure 2: LIG visualization for a LA sentence (Top) and LUA sentence (Bottom) of the SVO category. Green highlighted words contributed strongly towards the model output to be predicted as LA and LUA.

3 LIG for Constituency Parse Tree patterns

CPT is a type of grammar tree which captures the relations between the constituents of a sentence. We believe that analyzing the CPT patterns will give us insights into the grammatical structure of LA and LUA. Computing the LIG for CPT patterns at different subtree levels can give us an indication into the constituents which contribute largely towards making the sentence LA or LUA.

For each of the 5 category of sentences, we extracted all the CPT patterns for correctly classified sentences at every subtree level and picked the most frequent patterns at the root level (Table 2). The subtree patterns in bold are the highest ranking subtrees based on LIG. BERT has been shown to learn surface level features in the early layers, syntactic features in the middle layers and semantic features in the higher layers (Jawahar et al., 2019). Hence, we also wanted to analyse the LIG behaviour across the 12 layers and especially the early to middle layers which are relevant for grammar understanding. Across each of the categories in the LIG heatmaps, it was seen that the top subtree CPT patterns based on token embedding LIG were also dominating across the input and encoder layers of BERT and hence were also found in the top 10 patterns. Further, it can be observed in Figure 1 that there are specific subtrees which dominate more (shades of orange) as compared to others. This characteristic

Category	C	CC	MC	CC^+	CC^-	MC^+	MC^-	$CC^+\%$	$MC^+\%$
CIA	182	162	20	144	18	7	13	88.88	35
RAA	144	100	44	100	0	2	42	100	4.54
SVA	676	476	200	441	35	148	52	92.64	74
SVO	500	400	100	362	38	54	46	90.5	54
WHE	520	516	4	465	51	0	4	90.11	0

Table 3: LIG assessment for Correctly classified sentences (CC) and Misclassified (MC) sentences (C: Count, CC^+: Count of CC having positive LIG, CC^-: Count of CC having negative LIG, MC^+: Count of MC having positive LIG, MC^-: Count of MC having negative LIG, $CC^+\%$: Percentage of CC^+ in CC, $MC^+\%$: Percentage of MC^+ in MC).

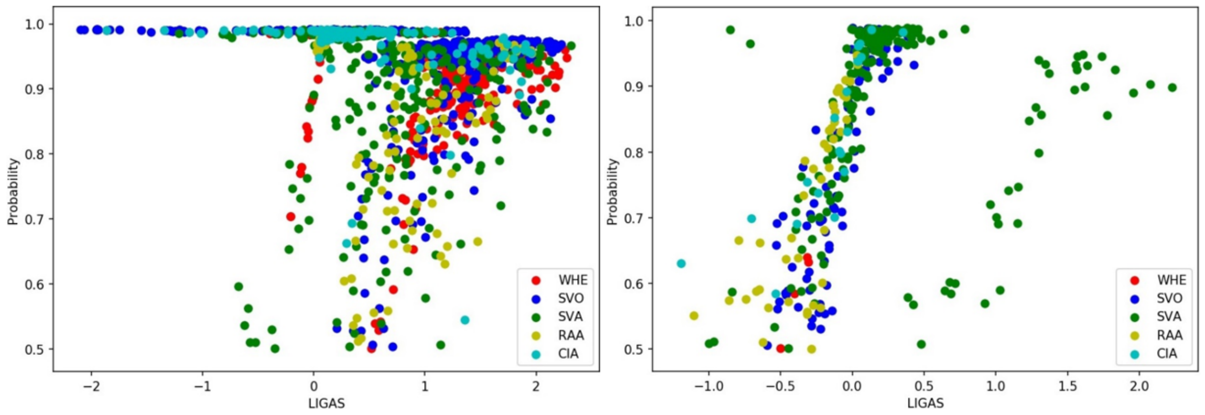


Figure 3: Prediction Probability vs LIG Scatter plots for Correctly classified (Left) and Misclassified (Right) sentences.

is especially useful for debugging LUA as it helps us to understand which phrase contributed largely towards making it unacceptable.

Further, it can be observed in Table 2 that the LIG for LUA are significantly larger than LA. The dominating CPT subtree patterns had a large spike in the LIG for LUA in comparison to LA, indicating that linguistically acceptable patterns were not being adhered. In Figure 2 we can see how the different words in the LA and LUA of the SVO category contributed in varying magnitudes towards the model's prediction.

4 LIG and Prediction confidence of the model

We investigated to check if there is a relationship between the LIG and the prediction confidence of the model. We found that the range of correctly classified sentences having positive LIG is between 88% to 100% ($CC^+\%$ in Table 3) indicating that whenever the input contributes strongly towards a particular class (whether it is LA or LUA), the model has a higher confidence in making the correct prediction. Around 43% (MC^- in Table 3) of the total misclassified sentences had negative LIG which showed that the features disagreed with the model's prediction. This behaviour can be observed distinctly in the Figure 3 scatter plots, where we notice that there a large number of points near the top right corner for the correctly classified sentences, and a large number of points near the bottom left corner in the case of misclassified sentences.

We believe that this indication can be used to improve the model's performance by parameterizing the LIG in the loss function during the later stages of the training process once the model has achieved a reasonable performance (to ensure that the gradients computed are meaningful) and hence serve as a correction mechanism for the model. This aligns with a previous work (Erion et al., 2021) which showed that axiomatic attribution priors improved model performance on many real-world tasks.

5 Conclusion

We have proposed a novel approach for explaining the Linguistic Acceptability criteria learnt by BERT using LIG and CPT patterns. As there is a strong relationship between LIG and the prediction confidence of the model, our future work will focus on parameterizing the LIG in the loss function and observing the model's performance.

References

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

Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. 2016. Layer-wise relevance propagation for neural networks with local renormalization layers. In *International Conference on Artificial Neural Networks*, pages 63–71. Springer.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. 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.

Gabriel Erion, Joseph D Janizek, Pascal Sturmfels, Scott M Lundberg, and Su-In Lee. 2021. Improving performance of deep learning models with axiomatic attribution priors and expected gradients. *Nature Machine Intelligence*, pages 1–12.

Allyson Ettinger, Ahmed Elgohary, and Philip Resnik. 2016. Probing for semantic evidence of composition by means of simple classification tasks. In *Proceedings of the 1st workshop on evaluating vector-space representations for nlp*, pages 134–139.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does bert learn about the structure of language? In *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*.

Katharina Kann, Alex Warstadt, Adina Williams, and Samuel Bowman. 2019. Verb argument structure alternations in word and sentence embeddings. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2019*, pages 287–297.

Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, et al. 2020. Captum: A unified and generic model interpretability library for pytorch. *arXiv preprint arXiv:2009.07896*.

Yongjie Lin, Yi Chern Tan, and Robert Frank. 2019. Open sesame: Getting inside BERT's linguistic knowledge. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 241–253, Florence, Italy. Association for Computational Linguistics.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. *arXiv preprint arXiv:1808.09031*.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153. PMLR.

Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. 2016. Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713*.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, et al.

2019. What do you learn from context? probing for sentence structure in contextualized word representations. *arXiv preprint arXiv:1905.06316*.

Shikhar Vashishth, Shyam Upadhyay, Gaurav Singh Tomar, and Manaal Faruqui. 2019. Attention interpretability across nlp tasks. *arXiv preprint arXiv:1909.11218*.

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 preprint arXiv:1706.03762*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Alex Warstadt, Yu Cao, Ioana Grosu, Wei Peng, Hagen Blix, Yining Nie, Anna Alsop, Shikha Bordia, Haokun Liu, Alicia Parrish, et al. 2019. Investigating bert's knowledge of language: Five analysis methods with npis. *arXiv preprint arXiv:1909.02597*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Ethan Wilcox, Peng Qian, Richard Futrell, Miguel Ballesteros, and Roger Levy. 2019. Structural supervision improves learning of non-local grammatical dependencies. *arXiv preprint arXiv:1903.00943*.

The Importance of Context in Very Low Resource Language Modeling

Lukas Edman **Antonio Toral** **Gertjan van Noord**
Center for Language and Cognition
University of Groningen
{j.l.edman, a.toral.ruiz, g.j.m.van.noord}@rug.nl

Abstract

This paper investigates very low resource language model pretraining, when less than 100 thousand sentences are available. We find that, in very low resource scenarios, statistical n-gram language models outperform state-of-the-art neural models. Our experiments show that this is mainly due to the focus of the former on a local context. As such, we introduce three methods to improve a neural model's performance in the low-resource setting, finding that limiting the model's self-attention is the most effective one, improving on downstream tasks such as NLI and POS tagging by up to 5% for the languages we test on: English, Hindi, and Turkish.

1 Introduction

With the advent of the Transformer (Vaswani et al., 2017) and masked language model (MLM) pretraining (Devlin et al., 2018), attention-based neural networks have proven quite effective at a variety of language tasks, provided that large amounts of data are available for pretraining. However, the performance can drop significantly as the number of sentences used for MLM pretraining decreases. This poses an issue for low-resource settings such as for underrepresented languages, where there is a limited amount of monolingual data.

Under low-resource conditions, attention-based models have difficulty learning from MLM, and as such statistical language models (SLMs) can outperform neural language models (NLMs). We demonstrate this by using a popular SLM toolkit, KenLM (Heafield, 2011), and test its accuracy on the MLM task compared to that of a Transformer model.[1] The results (Table 1) show that a trigram SLM is able to outperform the Transformer model by a wide margin for all languages when only 10 thousand sentences are available.

[1]The details of these tests are discussed in Section 3.1.

		Data Amount		
Language	System	10k	40k	100k
EN	NLM	12.8	30.7	**44.6**
	SLM	**29.7**	**37.9**	42.1
HI	NLM	27.0	**48.7**	**57.4**
	SLM	**45.7**	48.1	52.4
TR	NLM	6.4	22.3	36.2
	SLM	**23.1**	**30.5**	**39.9**

Table 1: English (EN), Hindi (HI), and Turkish (TR) MLM accuracy scores (%) for a neural versus statistical model.

While an SLM might outperform a neural model on MLM, the neural model has the benefit of being easily transferable to downstream tasks by means of fine-tuning. As such, this paper seeks to determine how we can improve the performance of an NLM to that of an SLM in low-resource scenarios. We investigate three approaches:

1. **Changing the input** by limiting the pretraining context size
2. **Changing the architecture** by limiting the self-attention window
3. **Changing the training objective** by using soft labels distilled from the SLM

We motivate and detail these methods in Section 2, describe experiment details in Section 3, show and discuss results in Section 4, and conclude our work in Section 5.

2 Methods

When comparing the general function of an SLM to an NLM, we consider the largest difference to be the context size considered. A tri-gram SLM will consider only the context of the adjacent two words on either side. For example, the score we use for word C in the sequence `A B C D E F G` is $\log(p(C|A,B) \times p(D|B,C) \times p(E|C,D))$.

Meanwhile, self-attention allows a Transformer to consider the entire context, which in XLM is 256 tokens by default (Lample and Conneau, 2019). Since XLM is trained with continuous streams of text, the input size is therefore always 256, and can consist of multiple sentences. In the low-resource setting, it may be difficult to learn important features from such a large context size.

To tackle this, we consider three alternative approaches. First, we put the strictest limitation on context by limiting the length of the input (§ 2.1). Second, we use a limited attention scope, thereby limiting the context within the first layer of the Transformer, but allowing information to flow from larger contexts in subsequent layers (§ 2.2). Finally, we put no explicit restriction on context size, but rather we expect the model to learn to limit itself via distillation from the limited statistical model (§ 2.3).

If context size is indeed the issue, we would expect the strictest form of limitation to perform best, as it would not need to learn to limit itself during training. This may be however too limiting for tasks which require a larger context, where we would expect that limiting attention would perform best. If context size is not the issue, we would expect that distilling knowledge from the statistical model would perform best, as its context is not limited, and the statistical model would still help the neural model learn a better strategy for language modelling than it is capable of on its own.

2.1 Changing the Input

We first limit the context size by presenting the input to a sliding window of a fixed context size. To stay consistent with the SLM, we only mask the middle word during MLM pretraining, padding the left and right side with BOS and EOS tokens respectively as needed.[2] For example, with a context size of 5 for the sentence "it is sunny today", we have:

```
[BOS] [BOS] [MASK] is sunny
[BOS] it [MASK] sunny today
it is [MASK] today [EOS]
is sunny [MASK] [EOS] [EOS]
```

This approach has the benefit of a smaller input complexity and an easier training objective (since only 1 word is masked at a time). These factors should make it easier for the model to learn the importance of local context. However, as the pretraining step does not expose the model to input longer than the context size, fine-tuning with a longer context size may hurt the model's performance.

2.2 Changing the Architecture

Rather than explicitly limiting the context, we also try limiting the model's attention towards words outside of the desired context. This is accomplished by adding a weight matrix to the query-key matrix produced during self attention. More specifically, referring to Equation 1 from Vaswani et al. (2017):

$$\mathrm{Attn}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V \qquad (1)$$

We add a band matrix W before applying softmax, where the elements within the band are 0 and the elements outside the band are $-\infty$, shown in Equation 2.[3] The size of the band corresponds to the context size (c), as the attention scores within the band are unaffected, whereas the attention outside of the band is effectively removed. This approach is very similar to that of the Longformer (Beltagy et al., 2020), which has a sliding-window attention with the aim of reducing model complexity and computation in long documents.

$$\mathrm{Attn}(Q, K, V) = \mathrm{softmax}(W + \frac{QK^T}{\sqrt{d_k}})V,$$
$$w_{i,j} = \begin{cases} -\infty & j < i - c \\ -\infty & j > i + c \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

While the model's self-attention range is limited to the defined context size, the ability for information from outside the context to associate with that of within the context is still possible in upper layers of the Transformer. For example, with a 6-layer encoder and a context size of 5, the first word could theoretically receive information about all words up to the 13th position. One benefit of this approach over limiting the input context (Section 2.1) is that the limitation can still be applied during fine-tuning.

2.3 Changing the Training Objective

The first two approaches have mainly been focused on the issue of context size, however the importance of an SLM having a fixed objective is not yet

[2]We also tried just limiting the context size without changes to MLM or the input, as done in contemporary work (Press et al., 2020), but the performance was worse.

[3]In practice we use -10^9.

addressed. While an NLM still has to learn its objective, we can potentially make this easier to learn by learning from the outputs of the SLM, inspired by knowledge distillation (Hinton et al., 2015).

In the MLM task, a model is typically trained to compare its output for masked tokens to a "hard label", where the probability is 1 for the actual word and 0 for all others. Rather than training with hard labels, we construct a soft label from the output of the SLM. This is done by using the SLM's score of the context with each candidate word replacing the mask.[4] The scores are first min-max normalized, then weighted, and finally scaled to unit length (so that the probabilities sum to 1).[5] The weighting is done by raising each score to the nth power, acting as a "hardness" parameter, where the most likely candidates approach 1 and the least likely approach 0 as n increases. We experiment with $n \in \{1, 2, 4, 6, 8\}$ and find $n = 6$ to give the best results.

3 Experimental Setup

We test each of our three methods on English (EN), Hindi (HI), and Turkish (TR). We train our models with XLM (Lample and Conneau, 2019), starting from a random initialization. We use the first 10 or 40 thousand sentences per language[6] from the WMT2007 NewsCrawl for English (following XLM), WMT2013 NewsCrawl for Hindi, and the WMT2016 NewsCrawl for Turkish.[7] For all of the tests, the data is tokenized with UDPipe (Straka and Straková, 2017),[8] truecased with Moses (Koehn et al., 2007), and 10 thousand BPE (Sennrich et al., 2015) joins are used.

The architecture behind our models is a 6-layer Transformer with 8 attention heads, an embedding dimension size of 1024, dropout set at 0.1, and GELU (Hendrycks and Gimpel, 2016) activation. For pretraining, we use a batch size of 32, and the Adam optimizer (Kingma and Ba, 2014), with a learning rate of 1e-4. We lower the learning rate to 2.5e-5 for the fine-tuning tasks. We use an early stopping criterion of no improvement in accuracy (MLM accuracy for pretraining, NLI or POS tag accuracy for fine-tuning) on the validation set for 20000 iterations, with a patience of 10. [9]

3.1 Measuring MLM accuracy

For our initial experiment showing the MLM accuracy of an SLM versus an NLM, we use a trigram KenLM model as our statistical model, and XLM (Lample and Conneau, 2019) as our neural model. Both KenLM and the XLM model are trained on the same 10 or 40 thousand sentences. Being a statistical model, KenLM's training process simply consists of tabulating frequencies, which are then used to estimate probabilities during inference.

As KenLM outputs scores for entire sequences, we simulate prediction of a masked word by replacing the word with every word in the vocabulary, and take its highest score as its prediction.[10] We repeat this for every word in the sentence for the first 100 sentences of the dataset,[11], producing roughly 2600 examples.[12]

3.2 Downstream Tasks

We fine-tune our models on the Natural Language Inference (NLI) task. For training, we use the MultiNLI dataset (Williams et al., 2018), and for development and testing, we use the XNLI dataset (Conneau et al., 2018).

When fine-tuning on XNLI for our limited attention model (Section 2.2), the first token (the `CLS` token used for classification) in the final layer often cannot access information from the second sentence. As such, we instead average every token rather than simply taking the first token, which improves results dramatically. We did not find this to improve any of our results with the other approaches, so we use only the first token in the other approaches.

[4]Each masked word is handled separately, so in a sentence with multiple masked words, the mask does not appear as part of the context for the SLM.

[5]To limit memory usage, scores below the top 100 are zeroed out after normalization.

[6]The datasets come pre-shuffled.

[7]`http://www.statmt.org/wmt16/translation-task.html`

[8]We use UDPipe so that the tokenization for our POS tagging data (which comes from UD) is consistent with the pretraining.

[9]As we used the XLM implementation from `https://github.com/facebookresearch/XLM`, any hyperparameters not mentioned are set at their default values.

[10]Because these scores are chain probabilities, it is not clear how to get a perplexity score comparable to that of an NLM, which is why we chose to compare with MLM accuracy. However the MLM accuracies of the NLMs follow the same trend as their perplexities.

[11]We use WMT `newstest2016` from English–German for English and English–Turkish for Turkish, and `newstest2014` for English–Hindi for Hindi.

[12]Unlike in standard MLM during training, for evaluation only one token is masked in a sentence at a time. Masking multiple tokens would increase the number of queries to the KenLM model exponentially.

We also investigate an easier task that typically requires less context, part-of-speech (POS) tagging, in appendix B. When applicable, the training data for both tasks is limited to the first 10 or 40 thousand sentences, according to the amount of data used in pretraining.

4 Results

We now compare the results of the SLM, normal NLM, and our 3 improvements to the NLM: limited context (NLM-C), limited attention (NLM-A), and the hybrid training objective (NLM-H). For NLM-C and NLM-A, we experiment with different context sizes and attention window sizes, ranging from 5 to 13. The SLMs are trigram models, and NLM-H uses these models for its soft labels.

4.1 Pretraining

Table 2 shows the MLM accuracies for all of the methods, using 10 and 40 thousand sentences. As we can see, the standard NLM is the worst, each of the 3 additions improve on the standard NLM, with NLM-C performing similarly to the SLM.

			10k			40k	
System	Context	EN	HI	TR	EN	HI	TR
NLM	256	12.8	27.0	6.4	30.7	48.7	22.3
NLM-C	5	27.4	45.1	22.4	37.5	50.1	31.7
	9	28.1	45.9	22.6	39.3	**53.3**	**32.8**
	13	29.4	**46.2**	22.8	**40.4**	52.9	31.0
NLM-A	5	23.7	41.7	17.1	36.9	51.5	30.4
	9	21.5	42.6	11.4	37.6	51.3	29.7
	13	20.1	42.6	10.3	37.6	51.3	27.7
NLM-H	256	22.7	38.9	14.1	33.1	48.8	27.6
SLM	5	**29.7**	45.7	**23.1**	37.9	48.1	30.5

Table 2: MLM accuracies (%), best in bold. The "Context" column refers to the attention window for NLM-A, and the input size for the others.

The similarity in performance for NLM-C and SLM strongly suggests that local context is the most important factor in SLM's outperformance over NLM. This focus on local context also has an impact on the performance of rare words, as the NLM specifically fails to fill in the mask when the masked word is a word from the 80% least frequent words. We discuss this in detail in appendix A.

NLM-A and NLM-H also outperform NLM, but not to the degree of NLM-C. While NLM-A has a similar goal as NLM-C, the degree to which information can flow from a wider context may be inhibiting the model from focusing on local context. This would explain why the accuracies decrease as the attention window increases. For NLM-H, since the context is not explicitly limited, it can similarly suffer from the complexity of self-attention.

4.2 NLI

Natural Language Inference (NLI), involves classifying two statements into three classes: "contradiction", "entailment", and "neutral". This typically would require a large context as the relation between the two sentences' meanings needs to be understood. As our focus for two of our approaches was to limit their context, we would expect this task to be the most challenging. Our results are in Table 3.

			10k			40k	
System	Context	EN	HI	TR	EN	HI	TR
NLM	256	45.6	41.5	42.0	53.2	49.8	49.4
NLM-C	5	44.0	42.2	42.1	51.8	47.4	46.9
	9	44.8	43.2	42.4	51.8	47.0	46.5
	13	45.2	42.5	41.4	50.1	47.2	46.5
NLM-A	5	43.4	44.5	40.5	53.6	48.2	47.9
	9	46.8	45.1	44.6	**54.4**	**50.2**	**50.2**
	13	**46.9**	**46.8**	**45.8**	54.2	49.7	**50.2**
NLM-H	256	45.0	42.1	44.8	52.6	49.4	49.2

Table 3: NLI accuracies (%), best in bold.

The results on NLI differ greatly from the MLM accuracies, as NLM-A performs the best across the board, despite its MLM accuracy being lower than NLM-C (cf. Table 2). This is likely due to NLM-A needing no changes to the input between the pretraining and fine-tuning steps. Meanwhile, NLM-C performs more poorly as it needs to adjust to the longer input for fine tuning.

When comparing the context sizes, we see that a larger context size in general performs better. This is in line with the idea that NLI generally demands a larger context size.

5 Conclusion

Despite the ubiquity of pre-trained neural language models (NLMs) in state-of-the-art NLP, in the low-resource setting they are outperformed by statistical language models (SLMs). Their general formulation assumes a large amount of data for pretraining, so in this work we adapt them to better perform in low-resource conditions.

We found that the complexity of self-attention on large contexts is a major inhibitor. As a solution to this, we propose shortening the attention span (NLM-A), which we show can increase the model's performance on downstream tasks. We believe that an ideal limitation of attention span would be initially very limited, but the span would increase dynamically during training. We plan to look into this further in future work.

For the best performance on MLM accuracy during pretraining itself, we propose limiting the size of the input (NLM-C), improving upon the standard method for training neural models. This achieves SLM-level performance on the lowest resource setting (10 thousand sentences), and outperforms an SLM on slightly higher-resource settings (40 thousand sentences). In addition, the neural model with a limited context can, unlike the SLM, be transferred to downstream tasks.

While limiting the input size (NLM-C) performs better than limiting the attention span (NLM-A) for pretraining, the opposite is the case for downstream tasks. As a potential solution for this, we propose for future work a second pretraining step in which the non-limited input is used.

Finally, our work primarily serves to investigate how attention-based models function with very little data. However in many real-world scenarios, transfer learning from large multilingual models is often used. Looking at the impact of these methods with multilingual transfer learning employed alongside is something we plan to do in the future.

References

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. *arXiv preprint arXiv:1809.05053*.

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*.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the sixth workshop on statistical machine translation*, pages 187–197.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Ofir Press, Noah A. Smith, and Mike Lewis. 2020. Shortformer: Better language modeling using shorter inputs.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. 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. *arXiv preprint arXiv:1706.03762*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. 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 1112–1122. Association for Computational Linguistics.

Daniel Zeman, Joakim Nivre, Mitchell Abrams, et al. 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.

A Pretraining Analysis

To better understand the failures of the NLM model on MLM accuracy, we look at the performance of our models with respect to the frequency of each word during training. We split the vocabulary into 5 equal bins according to frequency and record the accuracy on those bins, shown in Table 4.

	Bin	NLM	NLM-A	NLM-C	NLM-H	SLM
10k	1	0.0	1.2	14.3	2.0	**16.1**
	2	0.0	2.9	9.2	3.0	**11.5**
	3	0.1	3.1	9.0	4.1	**10.8**
	4	0.0	3.6	9.9	3.2	**14.8**
	5	15.4	27.1	32.2	25.7	**34.2**
40k	1	5.7	14.8	**17.9**	15.2	**17.9**
	2	7.5	15.4	10.0	14.6	**19.2**
	3	9.9	17.7	19.0	14.6	**24.6**
	4	8.5	16.4	17.4	13.1	**19.9**
	5	33.5	39.5	**43.3**	36.2	42.7

Table 4: Accuracy (%) per frequency bin for English, with bin 1 being the least frequent 20%, and bin 5 being the most frequent 20%. For NLM-A and NLM-C, we only report the scores for the systems with a context size of 5.

The SLM performs better across the board, but the NLM specifically fails on the least common 80% of words when 10 thousand sentences are used. While less frequent, this still accounts for roughly 20% of the words seen in training data, so the impact is understandably substantial. Interestingly, NLM-C performs similarly to SLM, which reinforces the idea that context size is the main reason why SLMs outperform standard NLMs in the low resource setting.

We also attempt to measure the "reasonableness" of a system's guess for MLM. Considering words split into multiple tokens by BPE, we measure how often the system completes them to a word that is in the vocabulary. For example "up@@" could be reasonably completed with "grade" or "date". As the meaning of an entire sentence is not considered, local context is especially important for completing this task. We show the results in Table 5.

		NLM	NLM-A	NLM-C	NLM-H	SLM
10k	EN	2.2	22.8	52.8	33.9	**61.1**
	TR	4.4	32.2	**42.2**	32.8	39.9
40k	EN	40.3	57.3	69.7	57.0	**78.7**
	TR	45.3	54.7	55.0	51.9	**55.1**

Table 5: Word completion (%) for English and Turkish. Showing systems with context 5 for NLM-A and NLM-C.

The results show a drastic difference in performance of NLM to SLM when trained on 10 thousand sentences. The standard NLM seems to fail to understand the concept of multi-token words. NLM-C and SLM again perform similarly. Interestingly, the discrepancy in performance on the two languages for the SLM is larger than for the NLMs. While this not central to the topic of this paper, it may be worth exploring it further.

Despite performing well on the downstream tasks, NLM-A does not perform particularly well on these pretraining metrics. This may showcase the inherent difficulty in evaluating the quality of the pretraining objective, as metrics like MLM accuracy or word completion do not give a clear indication of the transferability of a pretrained model to a downstream task.

B POS Tagging

Part-of-speech (POS) tagging is considered a much easier task than NLI, as most words do not need a large amount of context to be tagged. This should be an ideal setting for the context-limited methods to perform well, particularly NLM-C.

We use the POS tagging data from Universal Dependencies (UD) v2.7 (Zeman et al., 2020), using the English-GUM and Turkish-BOUN datasets.

The results on POS tagging (Table 6) are somewhat similar to the NLI results, as NLM-A again performs the best. As this task is more suited for the contextually-limited NLM-C, we would expect it to perform similarly well, however this is not the case. We believe NLM-C's poor performance can again be attributed to the increase in context size for fine-tuning.

		10k		40k	
System	Context	EN	TR	EN	TR
NLM	256	89.2	87.5	92.8	88.9
NLM-C	5	90.8	87.2	91.7	87.7
	9	90.5	87.7	92.1	88.4
	13	90.7	88.3	92.2	88.2
NLM-A	5	**92.5**	**89.2**	94.2	90.0
	9	**92.5**	**89.2**	93.9	**90.1**
	13	91.6	88.5	**94.3**	90.0
NLM-H	256	91.4	88.3	93.1	88.9

Table 6: POS tagging accuracies (%), best in bold.

The importance of local context for the POS tagging task is highlighted by the scores of NLM-A and NLM-C, where overall the models with a smaller context perform better than those with a larger context. NLM-H however does still provide improvements over the standard NLM, which may indicate that the network can more easily learn to limit its self-attention from the soft labels.

Stylistic MR-to-Text Generation Using Pre-trained Language Models

Kunal Pagarey, Kanika Kalra, Abhay Garg, Saumajit Saha, Mayur Patidar, Shirish Karande
TCS Research Pune, India
{kunal.pagarey, kalra.kanika, abhay.garg1, saha.saumajit, patidar.mayur, shirish.karande}@tcs.com

Abstract

We explore the ability of pre-trained language models BART, an encoder-decoder model, GPT2 and GPT-Neo, both decoder-only models for generating sentences from structured MR tags as input. We observe best results on several metrics for the YelpNLG and E2E datasets. Style based implicit tags such as emotion, sentiment, length etc., allows for controlled generation but it is typically not present in MR. We present an analysis on YelpNLG showing BART can express the content with stylistic variations in the structure of the sentence. Motivated with the results, we define a new task of emotional situation generation from various POS tags and emotion label values as MR using EmpatheticDialogues dataset and report a baseline. Encoder-Decoder attention analysis shows that BART learns different aspects in MR at various layers and heads.

1 Introduction

Recent advances in NLG focus on generating text from structured data encoded as Meaningful Representations (MR). MR typically comprises of semantic content to be realized for generation. This can be used for automating writing reports from tabular data, descriptions and reviews for products or restaurants from catalog, etc. However, style based implicit tags can add dynamic, engaging and immersive effect in real world NLG applications such as social and empathetic chatbots. The style aspects along with content information allows generating varied and customized text with same content. In this work, we explore capabilities of an encoder-decoder model, BART (Lewis et al., 2019), and two decoder-only models, GPT2 (Radford et al., 2019) and GPT-Neo (Black et al., 2021) for MR-to-text generation task. We evaluate BART, GPT2 and GPT-Neo on three datasets, one for content and other for both content and style. These datasets include E2E original and clean version (Dušek et al., 2020) (Dušek et al., 2019) which are restaurant description datasets comprising of content based MR and Yelp NLG (Oraby et al., 2019), a restaurant reviews corpus having both semantic and stylistic tags. We define a new task of emotional situation generation on Empathetic dialogues (ED) dataset (Rashkin et al., 2018). We construct MRs using set of POS tag (Qi et al., 2020) values from situation along with emotion label. Table 9 of Appendix A describes sample input MR and output for each dataset.

Our main contributions are defined as: a) The ability of encoder-decoder based and decoder-only pretrained transformer models to generate fluent sentences from content and style based MR. b) A new task on emotional situation generation using POS tag and emotion label values as MR and report its baseline. c) Encoder-Decoder attention map analysis of BART to further understand which layer and head learns which concept.

2 Related Work

Existing structured data to text datasets - E2E (Dušek et al., 2020) (Dušek et al., 2019), WebNLG (Gardent et al., 2017), TOTTO (Parikh et al., 2020), AGENDA (Koncel-Kedziorski et al., 2019) etc consider input in various formats such as slot value pair, triplets, or graph. They consist of content based semantic input in MR. Recently introduced YelpNLG dataset by (Oraby et al., 2019) considers style aspect in addition to content slot value in MR and provides LSTM encoder decoder baseline. Our work focuses on exploring recent language model capability for content and style based MR.

Researchers have attempted to improve content slot value MR to text in attention based encoder decoder architectures by incorporating various techniques. (Tseng et al., 2020) performed joint training of NLU and NLG. (Roberti et al., 2019) in-

Dataset	Size	Content	Style
Yelp	300k	restaurant[],cuisine[], food[], staff[], service[], ambiance[], price[]	Sentiment(positive, negative, neutral), length(short,medium,long), perspective(first,person, not first person),exclamation (has exclamation, no exclamation)
E2E	50k	name[], eatType[],food[],near[], priceRange[], customerRating[], area[], kidsFriendly[]	NA
ED	25k	POS values subset from [Noun], [Adjective], [Verb], [Pronoun]	32 Emotion Labels

Table 1: Content and style tag description for each dataset. ED only consists of values without content slot type.

troduced copy mechanism from MR facts to text. (Kedzie and McKeown, 2020) performed controllable MR-to-text generation by comparing different linearization strategies and phrase-based data augmentation technique. (Juraska et al., 2018), (Zhang et al., 2018), (Gong, 2018) applied re-ranking on top of seq2seq model providing semantic control, (Puzikov and Gurevych, 2018) came up with data-driven and template-based generation system. (Shen et al., 2019) used computational pragmatic based approach for conditional generation. However, we observe that all pre-trained transformer models perform well irrespective of their sizes, without requiring changes for both content and style MR.

3 Dataset Description

Table 1 provides a concise description of the datasets used, which were constructed to explore and improve the natural language generation capability of neural architectures. E2E (Dušek et al., 2020) original, a restaurant review dataset, has high lexical diversity and diverse discourse phenomena. E2E clean by (Dušek et al., 2019) is a noise free version of E2E (Dušek et al., 2020), with no mismatch between the content of the MR tags and the corresponding references. (Oraby et al., 2019) curated MR for YelpNLG automatically by leveraging freely available user review data on restaurants. This dataset brings in rich language descriptions with varied semantic emotions and content. To further explore the empathetic conversational potential, we use ED dataset (Rashkin et al., 2018), which comprises emotional dialogues between two persons. Motivated by YelpNLG, we constructed MR using values from POS tag set from noun, adj, pronoun and emotion label values for emotional situations provided in ED dataset.

4 Experiments

We fine-tune pre-trained language models like BART-large, GPT2-medium and GPT-Neo 125M for MR-to-text on train split of respective datasets. We use early stopping and choose the best model for evaluation on test set. Other parameters used for fine tuning are AdamW optimizer with a learning rate of 3e-5 and a linear learning rate scheduler. While generating the output text from MR we use beam search decoding with beam size of 4. We evaluate the generated output text using the standard automatic evaluation metrics[1] BLEU (Papineni et al., 2002), METEOR (Lavie and Agarwal, 2007), NIST (Doddington, 2002), CIDEr (Vedantam et al., 2015) and ROUGUE (Lin, 2004), and Semantic Error Rate (SER) (Dušek et al., 2019).

5 Results and Discussion

E2E, E2E clean: We report the fine-tuning results of the pre-trained models for E2E in Table 2 and compare it with other recent baselines[2]. We obtain best METEOR, CIDEr and ROUGE-L scores for E2E original using GPT2 and for E2E clean, best NIST using GPT2 and best SER score using BART. The other scores are comparable with baselines and do not differ significantly. The results show that the pre-trained models are able to preserve the content tags in output.

YelpNLG: We report the results for Yelp NLG in Table 3. We consider all the settings for YelpNLG - only content (BASE), content with style addition at different granularity (adjectives, sentiment, all other style aspects) . We obtain best results on all the metrics excluding SER using BART. SER less than 5% for BASE and STYLE setting signifies that

[1]`https://github.com/tuetschek/e2e-metrics`

[2]We show few baseline scores due to space constraint.

	Architectures	BL(↑)	NT(↑)	MT(↑)	RL(↑)	CD(↑)	SER(↓)
E2E Original	(Dušek and Jurčíček, 2016)	0.6593	8.6094	0.4483	0.685	2.2338	3.56*
	(Zhang et al., 2018)	0. 6545	8.184	0.4392	0.7083	2.1012	-
	(Tseng et al., 2020)	0.6855	-	-	-	-	-
	(Shen et al., 2019)	**0.6860**	**8.73**	0.4525	0.7082	2.37	-
	BART	0.6757	8.7242	0.4614	0.703	2.3914	3.58
	GPT2	0.6853	8.7164	**0.4637**	**0.7143**	**2.411**	5.56
	GPT-Neo	0.6841	8.6654	0.4626	0.7064	2.3697	**3.52**
E2E Clean	(Dušek and Jurčíček, 2016)	0.4073	6.1711	0.3776	0.5609	1.8518	0.87
	(Harkous et al., 2020)	**0.436**	-	**0.39**	**0.575**	**2.0**	-
	BART	0.4258	6.4188	0.3858	0.5677	1.9355	**0.13**
	GPT2	0.4285	**6.4524**	0.3854	0.5718	1.9873	1.02
	GPT-Neo	0.4087	6.2472	0.3751	0.5561	1.7928	4.06

Table 2: Results on E2E original & Clean test set. * - score on provided outputs. All tables follow these abbreviations - BL: BLEU, NT: NIST, MT: METEOR, RL: Rouge-L, CD: CIDEr

	Variant	BL	MT	CD	NT	SER
Baseline	Base	0.126	0.206	1.300	3.840	0.053
	+Adj	0.164	0.233	1.686	4.547	0.063
	+Sent	0.166	0.234	1.692	4.477	0.064
	+Style	0.173	0.235	1.838	5.537	0.090
Bart	Base	**0.177**	**0.227**	**1.820**	**5.303**	0.0346
	+Adj	**0.224**	**0.263**	**2.355**	**6.130**	0.0358
	+Sent	**0.225**	**0.264**	**2.358**	**6.158**	0.0382
	+Style	0.226	**0.268**	**2.587**	6.143	0.0435
Gpt2	Base	0.1673	0.2235	1.7731	4.7605	**0.0291**
	+Adj	0.2057	0.2578	2.2868	4.8509	**0.0308**
	+Sent	0.2072	0.2594	2.2971	4.8365	**0.0302**
	+Style	**0.2276**	0.2648	2.5799	**6.3915**	**0.0337**
GptNeo	Base	0.1646	0.2181	1.6502	5.052	0.0345
	+Adj	0.1972	0.2546	2.2095	4.6360	0.0320
	+Sent	0.2006	0.2548	2.2104	4.8331	0.0315
	+Style	0.2223	0.2611	2.5034	6.3503	0.0443

Table 3: Results on YelpNLG test set. Base MR only contains content slot type-value pairs, +Adj contains content slot type-value-adjective triplets. In addition to +Adj, sentiment and other stylistic aspects are added in +Sent and +Style, respectively.

	Variant	BL	MT	CD	NT	RL
Bart	NAd	**0.245**	**0.293**	**2.414**	**6.939**	**0.539**
	NAdP	**0.358**	**0.349**	**3.638**	**8.383**	**0.660**
Gpt2	NAd	0.1855	0.2589	1.8918	5.9274	0.4852
	NAdP	0.2726	0.3071	2.8072	7.2803	0.5994
GptNeo	NAd	0.1389	0.2392	1.6136	4.6187	0.4474
	NAdP	0.2263	0.2925	2.4268	6.8125	0.5829

Table 4: Results on ED test set. NAd: noun+adjective, NAdP: NAd+pronoun

the pre-trained models understand and express the content specifications well in the generated review. The models learn to associate the attribute values in the MR tag even in presence of different stylistic aspects in the fairly complex sentences.

ED: We report the results for ED dataset in Table 4. For ED, we provide baseline for various sequences of POS tag values in MR - (Noun,Adj), (Noun,Adj,Pronoun), with emotion label. We also find here that BART performs best compared to other pretrained models. We observe that increasing the content value information leads to increment in scores. We observe that the emotional aspect of the generated statements can be manipulated by changing the input tags, which emphasizes the models' power to generate customized sentences while expressing all the relevant content as shown in Table 5 and Table 6.

5.1 Attention Map Analysis for YelpNLG

To further investigate the rationale behind the superior performance of the transformer models, we extend (Vig, 2019) to analyse the encoder decoder attentions of BART and report results in Table 7. As in CNNs, we hypothesised that different nodes in the architecture would learn to capture particular tags in the MR. Thus, we evaluate which head and layer learns to attend to a particular style aspect. LxHy signifies encoder-decoder attention map corresponding to yth head of xth layer. We take 1000 samples in each case and perform element wise summation of attention distribution scores of the chosen output tokens shown under 'By' over all input tokens. We report the percentage of cases in which the input token under 'To' was in top3

YelpNLG MR	Output
food||chicken||delicious||mention_1 food||fries||no_adj||mention_1,positive,len_medium,not_first_person,no_exclamation	the chicken was delicious and the fries were cooked to perfection.
food||chicken||delicious||mention_1 food||fries||no_adj||mention_1,positive,len_medium,not_first_person,has_exclamation	the chicken is delicious and the fries are to die for!
food||chicken||tasty||mention_1 food||fries||delicious||mention_1,negative,len_long,not_first_person,has_exclamation	the chicken was tasty, the fries were delicious, but the whole thing was a let down!
food||fries||tasteless||mention_1 food||chicken||yuck||mention_1,positive,len_medium,first_person,has_exclamation	the fries were tasteless, the chicken was yuck, but i love this place!

Table 5: Generated outputs for YelpNLG using BART.

ED	Output
joyful(friends different country)	My friends are coming to visit me from a different country.
guilty(friends different country)	I lied to my friends that I was going to visit them in a different country.
anxious(husband end month little worried funds)	My husband and I are going overseas at the end of the month. I am a little worried about how we will manage the funds.
Confident(husband end month little worried funds)	My husband and I are going to get married at the end of the month. I'm a little worried about the funds we'll have, but I know we'll make it happen.

Table 6: Generated outputs for ED for Emotion with Noun and Adjective using BART.

(top5 for 'all') most maximally attended input tokens. 'Sample Type' column denotes the common stylistic aspect for all those samples. We observe that various layers and heads learn different stylistic concepts beyond just learning to copy (as shown in Figure 1, 2, 3 and 4 of Appendix C). The results reinforce our hy pothesis and establish that different parts of BART learn to understand the intrinsic meaning of different tags.

5.2 Qualitative Analysis

The generated outputs emphasize sensitivity of BART towards stylistic aspects. A minute change from no_exclamation in first row to has_exclamation in second row in Table 5, to our surprise, BART has generated very different and dramatic output. The last two rows show BART's capability of handling contrasting scenarios wherein the sentiment of the input is in contrast to the adjective values of food. Results in Table 6 show that BART can express the same content with different emotions fed as implicit tags.

While analysing the predictions of the finetuned models, we observe that most of the time, BART has been successful in generating output as per the given style constraints mentioned in the input MR tags. Table 8 shows a few instances where BART, out of all the 3 models, is capable of producing better and more meaningful sentences. However in the last two examples of Table 8, we also report cases where these models produce incoherent outputs that do not meet the required stylistic aspects.

LH	Token Maximally Attended (To,By)	%	Sample Type
L7H0	('negative', all)	86	negative
L7H0	('neutral', all)	60.5	neutral
L1H11	('first', first person words)	100	first person
L3H8	('short', all)	99.4	short
L8H5	('has', '!')	87.7	has exclamation
L7H11	(adjectives, food values)	73	adjs
L0H15	(food values, food values)	100	food values

Table 7: Attention Analysis for YelpNLG.

5.3 Analysis of the Pre-trained models

We have used pretrained models like BART-large, GPT2-medium and GPTNeo-125M in our experiments. The number of parameters used for these models vary. For example, BART-large has 406 million parameters, GPT2-medium has 345 million parameters and GPTNeo-125M has only 125 million parameters. Table 2, 3, 4 show that GPTNeo-125M, inspite of having less than 50% of the param-

MR: food||steak||awesome||mention_1 food||chicken||fried||mention_1,neutral,len_long,first_person, no_exclamation
Ref: "what i had in mind, try out their awesome steak and their fried chicken... however, it turns out they are only served on weekdays."
Bart: "the steak was awesome, but the fried chicken was a bit too sweet for my taste buds, but it was still good."
Gpt2: "i had the steak which was awesome, and my friend had the fried chicken which was also good."
GptNeo: "the steak was awesome, but the fried chicken was a little dry for my taste."

MR: food||wontons||no_adj||mention_1 food||wontons||no_adj||mention_2 food||meat||no_adj||ment ion_1,positive,len_medium,not_first_person,no_exclamation
Ref: "if you go anywhere in california, wontons are wontons with meat inside."
Bart: "the wontons are not just wontons, they are filled with meat and veggies."
Gpt2: "wontons are wontons, meat is meat."
GptNeo: "wontons, fried wontons, fried wontons, fried wontons, fried wontons, fried wontons, fried wontons, fried wontons, fried wontons."

MR: food||chicken||lemon||mention_1 food||vegetables||no_adj||mention_1,negative,len_long,not_first _person,no_exclamation
Ref: "first the appetizer was not prepared properly, out of the lemon chicken and then vegetables were cold."
Bart: "the lemon chicken had a lot of vegetables in it, but not enough of them to make up for the lack of taste."
Gpt2: "the lemon chicken had a lot of vegetables in it, but it was very dry."
GptNeo: "the lemon chicken had a lot of vegetables in it, but it was a little dry."

MR: food||chicken||no_adj||mention_1 food||sauce||no_adj||mention_1,negative,len_medium,first_per son,no_exclamation
Ref: i did not like the way the chicken was cooked or the sauce on it.
Bart: i had the chicken with the sauce on the side and it was good.
Gpt2: i had the chicken with the sauce on the side.
GptNeo: i had the chicken with the sauce on the side.

MR: food||pizza||no_adj||mention_1 food||steak||no_adj||mention_1 food||cheese||no_adj||ment ion_1,positive,len_medium,not_first_person,no_exclamation
Ref: love the pizza the steak and cheese and the place.
Bart: "pizza, wings, steak, cheese, and more."
Gpt2: "pizza, steak, cheese, etc."
GptNeo: "pizza, steak, and mac and cheese."

Table 8: Example outputs for YelpNLG using BART, GPT2 and GPTNeo.

eters of the other models, is being able to produce competitive results on these tasks, often beating the baseline performances in several cases.

GPTNeo has been pretrained on Pile (Gao et al., 2020) dataset, which is composed of different constituent sub-datasets from diverse domains. However, GPT2 and BART are pretrained exclusively on text data. The size of the pre-training dataset seems to have an impact in the performance of the pre-trained model on downstream tasks. This is because GPTNeo is trained on 800GB Pile dataset while GPT2 has been trained on only 40GB of webtext data.

6 Conclusion

We describe the benefits and importance of MR2text generation. We fine-tune recently introduced Transformer-based language models like BART, GPT2 and GPTNeo, and produce results on two versions of E2E, YelpNLG and ED datasets. We have defined a new task on Emphatetic Dataset to emphasize the usefulness of implicit tags in NLG. Quantitative and Qualitative analyses show how well BART captures the specifications and brings stylistic variations in generated outputs.

References

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large scale autoregressive language modeling with mesh-tensorflow.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145.

Ondřej Dušek, David M. Howcroft, and Verena Rieser. 2019. Semantic noise matters for neural natural language generation. In *Proc. of the 12th International Conference on Natural Language Generation*, pages 421–426, Tokyo, Japan. Association for Computational Linguistics.

Ondřej Dušek and Filip Jurčíček. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. *arXiv preprint arXiv:1606.05491*.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2020. Evaluating the State-of-the-Art of End-to-End Natural Language Generation: The E2E NLG Challenge. *Computer Speech & Language*, 59:123–156.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The webnlg challenge: Generating text from rdf data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133.

Heng Gong. 2018. Technical report for e2e nlg challenge. *E2E NLG Challenge System Descriptions*.

Hamza Harkous, Isabel Groves, and Amir Saffari. 2020. Have your text and use it too! end-to-end neural data-to-text generation with semantic fidelity. *arXiv preprint arXiv:2004.06577*.

Juraj Juraska, Panagiotis Karagiannis, Kevin K Bowden, and Marilyn A Walker. 2018. A deep ensemble model with slot alignment for sequence-to-sequence natural language generation. *arXiv preprint arXiv:1805.06553*.

Chris Kedzie and Kathleen McKeown. 2020. Controllable meaning representation to text generation: Linearization and data augmentation strategies. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5160–5185, Online. Association for Computational Linguistics.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text generation from knowledge graphs with graph transformers. *arXiv preprint arXiv:1904.02342*.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the second workshop on statistical machine translation*, pages 228–231.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Shereen Oraby, Vrindavan Harrison, Abteen Ebrahimi, and Marilyn Walker. 2019. Curate and generate: A corpus and method for joint control of semantics and style in neural nlg. *arXiv preprint arXiv:1906.01334*.

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.

Ankur P Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. Totto: A controlled table-to-text generation dataset. *arXiv preprint arXiv:2004.14373*.

Yevgeniy Puzikov and Iryna Gurevych. 2018. E2e nlg challenge: Neural models vs. templates. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 463–471.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2018. Towards empathetic open-domain conversation models: A new benchmark and dataset. *arXiv preprint arXiv:1811.00207*.

Marco Roberti, Giovanni Bonetta, Rossella Cancelliere, and Patrick Gallinari. 2019. Copy mechanism and tailored training for character-based data-to-text generation. In *Joint European Conference* *on Machine Learning and Knowledge Discovery in Databases*, pages 648–664. Springer.

Sheng Shen, Daniel Fried, Jacob Andreas, and Dan Klein. 2019. Pragmatically informative text generation. *arXiv preprint arXiv:1904.01301*.

Bo-Hsiang Tseng, Jianpeng Cheng, Yimai Fang, and David Vandyke. 2020. A generative model for joint natural language understanding and generation. *arXiv preprint arXiv:2006.07499*.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. *arXiv preprint arXiv:1906.05714*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Biao Zhang, Jing Yang, Qian Lin, and Jinsong Su. 2018. Attention regularized sequence-to-sequence learning for e2e nlg challenge. *E2E NLG Challenge System Descriptions*.

Deep Learning Based Approach For Detecting Suicidal Ideation in Code-Mixed Hindi-English: Baseline and Corpus

Kaustubh Agarwal
Computer Science and Engineering
Netaji Subhas University of Technology
New Delhi, India
kaustubh.co19@nsut.ac.in

Bhavya Dhingra
Electrical Engineering
Netaji Subhas University of Technology
New Delhi, India
bhavya.ee19@nsut.ac.in

Abstract

Suicide rates are rising among the youth, and the high association with suicidal ideation expression on social media necessitates further research into models for detecting suicidal ideation in text, such as tweets, to enable mitigation. Existing research has proven the feasibility of detecting suicidal ideation on social media in a particular language. However, studies have shown that bilingual and multilingual speakers tend to use code-mixed text on social media making the detection of suicidal ideation on code-mixed data crucial, even more so with the increasing number of bilingual and multilingual speakers. In this study we create a code-mixed Hindi-English (Hinglish) dataset for detection of suicidal ideation and evaluate the performance of traditional classifiers, deep learning architectures, and transformers on it. Among the tested classifier architectures, Indic BERT gave the best results with an accuracy of 98.54%.

1 Introduction

A study by the World Health Organization (WHO), has found that nearly 700,000 people die of suicide each year (WHO). Suicidal ideation, the act of thinking about, considering or planning suicide, can be attributed to multiple reasons including mental illness, traumatic stress, loss or fear of loss, social isolation, biological factors, environmental factors, genetic factors and situational factors (Wasserman et al., 2004). In the wake of COronaVIrus Disease-2019 (COVID-19), an increasing number of individuals across the world have become victims to one or more than one of these factors, that has led to increased rates of suicide worldwide (Fortgang et al., 2021).

It has been found that increase in consumption and posting on social media has a direct correlation to the tendency of expressing desires, thoughts, and intentions on pro-suicide platforms before attempting suicide (Gea and Sánchez, 2012). With COVID-19 driving social media consumption up by 72% and posting by 43% such incidents recorded an all-time high (Wold |, 2020). Danet and Herring (2007) mentioned that more than half of the people on social media platforms are not native English speakers and (Hong et al., 2011) confirmed that about 50% of the posts on Twitter are in languages other than English. These studies substantiate the need of a much broader scope for detection of suicidal ideation on social media than just the English language.

(Gupta et al., 2016) found that over 26% of the Indian population speaks more than one language, often in the form of code-switching or code-mixing. Code-switching occurs when an individual alternates between multiple languages in the context of a single conversation or situation while code-mixing is the use of two or more languages by an individual below clause level in a single social context. However, working with code-mixed data presents it's own set of challenges, including the creation of a large number of new constructions for understanding the syntax and semantics of the two or more combined languages, the availability of very small amounts of annotated data, and the use of drastically different approaches when compared to monolingual data (Çetinoğlu et al., 2016).

In this paper, we aim to detect suicidal ideation in code-mixed Hinglish. Although significant work is available for suicidal ideation detection in English (Castillo-Sánchez et al. (2020), Coppersmith et al. (2018), Mbarek et al. (2019), Ophir et al. (2020), Ramírez-Cifuentes et al. (2020), Sawhney et al. (2020), Shaoxiong Ji (2020), Tadesse et al. (2019), Vioules et al. (2018)), detection of suicidal ideation in code-mixed languages is relatively unexplored. To the best of our knowledge, we are the first to identify suicidal ideation in code-mixed

Table 1: Count and distribution of dataset

Category	Sample Count	Percentage
Suicidal Ideation (Positive Samples)	3098	47.41%
Non Suicidal (Negative Samples)	3435	52.59%

Hindi-English.

The contributions of our work include:

1. There is a significant lack of data in code-mixed suicidal ideation. We attempt to overcome this drawback by creating a dataset for suicidal ideation in code-mixed Hindi-English.
2. We propose the use of various existing models to create a baseline for future work in the field.

2 Methodology

The proposed methodology consists of three major parts, each consisting of a major contribution of our work.

2.1 Dataset Creation and Analysis

Even with the huge surge in suicidal ideation cases in code-mixed Hindi-English on social media platforms, there exists no dataset for suicidal ideation posts in it. Most existing research uses data from special Reddit channels like "Suicide Watch" (Ji et al. (2018), Tadesse et al. (2019)) or Twitter (Mbarek et al., 2019). However, since all of these datasets are in English, they fail to capture a large section of suicidal ideation texts that are unaccounted for due to medium of communication in specialized channels of social media (like subreddit "Suicide Watch"), frequent lack of hashtags, and deletion of such texts by social media companies due to the impact it can have on other users of their social media platform. To overcome a lack of a code-mixed dataset in this domain we scraped data from the subreddits such as Aww, Jokes, History, Discussion, Stories and Entertainment as negative samples and selected text from the "Suicide Watch" subreddit as positive samples. On the 6533 scraped samples thus obtained, we used the approach proposed by Gupta et al. (2020) to obtain code-mixed Hindi-English text from English text. The generated dataset consists of 6533 code-mixed text samples, 3098 of which are labelled as having suicidal ideation and 3435 labelled as having no suicidal ideation.

From Table-1 it can be observed that the data is fairly balanced. It is essential to ensure a fair representation of class labels in this context to eradicate unfounded bias due to training data.

Examples of the annotated data are:

Sample: Main literally aur figuratively khud ko maarnaa caahtaa huun
Translation: I want to kill myself, both literally and figuratively.
Label: Suicidal Ideation

Sample: Tumhara novels ya books mein favorite twist kaunsa hai?
Translation: What is your favorite twist in books or novels?
Label: Non Suicidal

The dataset is available on GitHub .

2.2 Creation of Hindi-English Bi-lingual Word Embeddings

Word embeddings are dense vectors that give semantic and syntactic information of words in a context (Mandelbaum and Shalev, 2016) and are a critical part of text classification tasks. However, creating embeddings requires a large amount of textual data. For this purpose, we have used 412k Hinglish tweets and 320k English tweets from Twitter for code-mixed Hindi-English data and preprocessed them by removing rare words, hashtags, mentions and Uniform Resource Locators (URLs). We experimented with two different experimental settings to form embeddings using two different techniques. For the first setting, we have used only the Hinglish tweets corpus to create embeddings and for the second one, a corpus of both English and Hinglish tweets combined. On each of these experimental settings, we tried the following two embeddings:

1. Word2Vec: This technique was introduced in 2013 (Mikolov et al., 2013) and is widely regarded as a pivotal method for creating dense word embeddings. Since a pre-trained corpus for English embeddings already exists, we trained our Hinglish corpus to create the required embeddings.

2. FastText: FastText was introduced by Facebook (Bojanowski et al., 2017a) as an extension of Word2Vec embeddings (Joulin et al., 2017). Instead of learning weights for words directly, FastText breaks words into multiple sub-words (Bojanowski et al., 2017b). This will be particularly helpful in representing rare words as embeddings since it is highly likely that their n-grams will be a subpart of other words. For example, “aww”, “awww” and it's variations, which are very common on social media platforms, can be trained appropriately.

2.3 Deep Learning Models

Four traditional classifiers, five deep learning classifiers and two transformers have been used to create a baseline. These models include:

1. Naive Bayes (Yu et al., 2015)
2. Random Forest (Breiman, 2001)
3. Linear SVM (Ladicky and Torr, 2011)
4. RBF Kernel SVM (Daqi and Tao, 2007)
5. Series CNN (Tang et al., 2021)
6. Parallel CNN (Yao et al., 2019)
7. LSTM (Hochreiter and Schmidhuber, 1997)
8. Bi-Directional LSTM (Schuster and Paliwal, 1997)
9. Attention Based Bi-Directional LSTM (Wang et al., 2016)
10. mBERT (Devlin et al., 2019)
11. Indic BERT (Kakwani et al., 2020)

All these architectures were presented with the task of binary classification where each text was predicted as a sample of suicidal ideation or a sample having no suicidal ideation.
While some of these models have been able to detect sarcasm, irony, and other factors that may affect the classification of a suicidal ideation text, its explicit learning and merging could be an avenue for future research.

3 Experimental Settings / Modeling

For training our deep learning models, we made a fifteen percent validation split for a total of 20 epochs while the transformers are trained for 5 epochs on the same split. The model checkpoints are saved at each epoch and the model with highest validation accuracy and lowest difference from training accuracy is saved as the final model to ensure prevention of overfitting.

Word embeddings are trained using negative sampling polarity, an embedding size of 300, a window length of 10. The Adam optimizer is employed in all of the models, coupled with the binary cross entropy loss function. With the exception of the output layer, which has sigmoid activation, all layers have relu activation. We tested CNN models with various kernel sizes, number of kernels, dropouts, and strides to see how well they performed. With the following parameters, the best results are obtained: stride = 1, number of kernels = 200, dropout = 0.5

For all RNNs the hyperparameters used are dropout for recurrent state = 0.25, dropout for input state = 0.25, and number of LSTM units = 400.

4 Results

We have tested our dataset on traditional machine learning classifiers, deep learning models and transformers. The results of well known traditional classifiers have been listed in Table 2. RBF Kernel SVM gives the best results among the traditional classifiers with an accuracy of 60.8% on the given corpus.

Table 2: Accuracy from traditional classifiers

Classifier	Accuracy
Naive Bayes	51.2%
Random Forest	55.7%
Linear SVM	59.2%
RBF Kernel SVM	60.8%

Deep Learning models are tested using Word2Vec and FastText embeddings on Hindi-English (Hinglish) data only, and on Hinglish and English data combined. Table 3 shows the results obtained by training deep learning models on this corpus. The Attention Bi-LSTM trained on a Word2Vec embedding of Hinglish and English data corpus gives the best result of 90.66%. It is observed that deep learning model architectures perform better with embeddings of Hinglish

Table 3: Accuracy from Deep Learning Models

	Hinglish Data		Hinglish + English Data	
Model	Word2Vec	FastText	Word2Vec	FastText
Series CNN	71.26%	71.24%	73.60%	73.12%
Parallel CNN	73.86%	73.86%	74.36%	74.16%
LSTM	79.64%	78.52%	81.72%	80.66%
Bi-LSTM	83.42%	82.64%	85.44%	84.74%
Attention Bi-LSTM	89.66%	87.42%	90.66%	89.82%

and English data combined instead of using just Hinglish data for creating embeddings which may be a result of better semantic and correlation coverage between embeddings on English data and Hinglish data. It's also worth noting that Word2Vec produces slightly better results than the FastText embeddings. This observation could be due to the fact that code-mixed data prevents n-grams from being used as the major classification criterion. Furthermore, the better performance of deep learning models over traditional classifiers can be attributed to the fact that they can learn more about human tendencies like sarcasm and irony (Sentamilselvan et al. (2021), Potamias et al. (2020)), thus reducing incorrect predictions on them.

Given the code-mixed nature of the corpus, the transformers used for classification are mBERT and IndicBERT. Table 4 shows the results of training our corpus on these transformers. IndicBERT slightly outperforms mBERT with an accuracy of 98.54%.

Table 4: Accuracy from Transformers

Classifier	Accuracy
mBERT	96.63%
Indic BERT	98.54%

The method of generation of the dataset could have influenced the results for mBERT's classification performance, however, it is highly unlikely as separate instances of mBERT have been used for each task and the tasks performed by them are highly specialized in the given scenario. The performance of IndicBERT on the same task proves the lack of apparent correlation between the semi-supervised technique used in the creation of the dataset (Gupta et al., 2020) and the classification accuracy of mBERT.

The problem of detecting suicidal thoughts on code-mixed Hindi-English data is compounded by a lack of clean data and linguistic complications connected with code-mixed data. More data, as well as well labelled classes, are necessary to allow the model to accept noise in textual input, spelling errors, diverse contexts, and stemming words.

5 Conclusions and Future Work

The current research is the first attempt to investigate multilingual text classification to predict suicidal ideation in code-mixed Hindi-English texts, and proposes a baseline for further work along with a corpus for validation. For the objective of detecting suicidal text on the created corpus, several deep learning based models are used, including CNN, LSTM, Bi-Directional LSTM, Attention Based Bi-Directional LSTM, mBERT and Indic BERT.Since texts containing suicidal ideation in Hinglish are not available directly, a dataset is created by using a semi-supervised approach to generate code-mixed Hinglish text using pre-trained encoders and transfer learning from anonymized data in English from Reddit.

A comparison of the various models indicated that both BERT-based models mBERT and Indic BERT give exceptional results and have accomplished the task with over 96% accuracy.

Multilingual text classification is still a developing field, and future advancements could lead to better outcomes. Comparing vectors aligned with multilingual word embeddings generated using MUSE to FastText pre-aligned word embeddings may generate better results. Working on factors such as sarcasm, humor and irony that affect the classification of suicidal ideation explicitly, along with their inclusion in the creation of the model could be another potential avenue for future research.

6 CRediT author statement

Kaustubh Agarwal: Conceptualization, Software, Formal Analysis, Investigation, Data Curation, Writing- Original Draft, Review & Editing.
Bhavya Dhingra: Project administration

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017a. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017b. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Leo Breiman. 2001. Random forests. 45(1):5–32.

Gema Castillo-Sánchez, Gonçalo Marques, Enrique Dorronzoro, Octavio Rivera-Romero, Manuel Franco-Martín, and Isabel De la Torre-Díez. 2020. Suicide risk assessment using machine learning and social networks: a scoping review. 44(12).

Özlem Çetinoğlu, Sarah Schulz, and Ngoc Thang Vu. 2016. Challenges of computational processing of code-switching. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 1–11, Austin, Texas. Association for Computational Linguistics.

Glen Coppersmith, Ryan Leary, Patrick Crutchley, and Alex Fine. 2018. Natural language processing of social media as screening for suicide risk. 10:117822261879286.

B. Danet and S.C. Herring. 2007. *The Multilingual Internet: Language, Culture, and Communication Online*. Oxford University Press.

Gao Daqi and Zhang Tao. 2007. Support vector machine classifiers using rbf kernels with clustering-based centers and widths. In *2007 International Joint Conference on Neural Networks*, pages 2971–2976.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Rebecca G. Fortgang, Shirley B. Wang, Alexander J. Millner, Azure Reid-Russell, Anna L. Beukenhorst, Evan M. Kleiman, Kate H. Bentley, Kelly L. Zuromski, Maha Al-Suwaidi, Suzanne A. Bird, Ralph Buonopane, Dylan DeMarco, Adam Haim, Victoria W. Joyce, Erik K. Kastman, Erin Kilbury, Hye-In S. Lee, Patrick Mair, Carol C. Nash, Jukka-Pekka Onnela, Jordan W. Smoller, and Matthew K. Nock. 2021. Increase in suicidal thinking during COVID-19. 9(3):482–488.

PM Gea and CB Sánchez. 2012. Psiquiatria.comSuicidio e Internet. Medidas preventivas y de actuación.

Deepak Gupta, Asif Ekbal, and Pushpak Bhattacharyya. 2020. A semi-supervised approach to generate the code-mixed text using pre-trained encoder and transfer learning. Association for Computational Linguistics.

Sakshi Gupta, Piyush Bansal, and Radhika Mamidi. 2016. Resource creation for hindi-english code mixed social media text.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. 9(8):1735–1780.

Lichan Hong, Gregorio Convertino, and Ed H. Chi. 2011. Language matters in twitter: A large scale study. In *Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17-21, 2011*. The AAAI Press.

Shaoxiong Ji, Celina Ping Yu, Sai fu Fung, Shirui Pan, and Guodong Long. 2018. Supervised learning for suicidal ideation detection in online user content. 2018:1–10.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

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*.

Lubor Ladicky and Philip Torr. 2011. Linear support vector machines. pages 985–992.

Amit Mandelbaum and Adi Shalev. 2016. Word embeddings and their use in sentence classification tasks.

Atika Mbarek, Salma Jamoussi, Anis Charfi, and Abdelmajid Ben Hamadou. 2019. Suicidal profiles detection in twitter. SCITEPRESS - Science and Technology Publications.

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, page 3111–3119, Red Hook, NY, USA. Curran Associates Inc.

Yaakov Ophir, Refael Tikochinski, Christa S. C. Asterhan, Itay Sisso, and Roi Reichart. 2020. Deep neural networks detect suicide risk from textual facebook posts. 10(1).

Rolandos Alexandros Potamias, Georgios Siolas, and Andreas Georgios Stafylopatis. 2020. A transformer-based approach to irony and sarcasm detection. *Neural Computing and Applications*, 32(23):17309–17320.

Diana Ramírez-Cifuentes, Ana Freire, Ricardo Baeza-Yates, Joaquim Puntí, Pilar Medina-Bravo, Diego Alejandro Velazquez, Josep Maria Gonfaus, and Jordi Gonzàlez. 2020. Detection of suicidal ideation on social media: Multimodal, relational, and behavioral analysis. 22(7):e17758.

Ramit Sawhney, Harshit Joshi, Saumya Gandhi, and Rajiv Ratn Shah. 2020. A time-aware transformer based model for suicide ideation detection on social media. Association for Computational Linguistics.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

K. Sentamilselvan, P. Suresh, G K Kamalam, S. Mahendran, and D. Aneri. 2021. Detection on sarcasm using machine learning classifiers and rule based approach. *IOP Conference Series: Materials Science and Engineering*, 1055(1):012105.

Shaoxiong Ji. 2020. Suicidal ideation detection in online social content.

Michael Mesfin Tadesse, Hongfei Lin, Bo Xu, and Liang Yang. 2019. Detection of suicide ideation in social media forums using deep learning. 13(1):7.

Wensi Tang, Guodong Long, Lu Liu, Tianyi Zhou, Jing Jiang, and Michael Blumenstein. 2021. Rethinking 1d-cnn for time series classification: A stronger baseline.

M. J. Vioules, B. Moulahi, J. Aze, and S. Bringay. 2018. Detection of suicide-related posts in twitter data streams. 62(1):7:1–7:12.

Yequan Wang, Minlie Huang, xiaoyan zhu, and Li Zhao. 2016. Attention-based LSTM for aspect-level sentiment classification. Association for Computational Linguistics.

Gail A. Wasserman, Susan J. Ko, and Larkin Mcreynolds. 2004. Assessing the mental health status of youth in juvenile justice settings. juvenile justice bulletin.

WHO. Suicide statistics.

Suzin Wold |. 2020. COVID-19 is changing how, why and how much we're using social media.

Hongdou Yao, Xuejie Zhang, Xiaobing Zhou, and Shengyan Liu. 2019. Parallel structure deep neural network using cnn and rnn with an attention mechanism for breast cancer histology image classification. *Cancers*, 11:1901.

Zhou Yu, Vikram Ramanarayanan, David Suendermann-Oeft, Xinhao Wang, Klaus Zechner, Lei Chen, Jidong Tao, Aliaksei Ivanou, and Yao Qian. 2015. Using bidirectional lstm recurrent neural networks to learn high-level abstractions of sequential features for automated scoring of non-native spontaneous speech. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 338–345.

On the Universality of Deep Contextual Language Models

Shaily Bhatt[1,2] [*]**, Poonam Goyal[1], Sandipan Dandapat[3], Monojit Choudhary[2], Sunayana Sitaram[2]**
[1] Birla Institute of Technology and Science, Pilani, India
[2] Microsoft Research, Bengaluru, India
[3] Microsoft R&D, Hyderabad, India
{f20170040, poonam}@pilani.bits-pilani.ac.in,
{sadandap, monojitc, sunayana.sitaram}@microsoft.com

Abstract

Deep Contextual Language Models (LMs) like ELMO, BERT, and their successors dominate the landscape of Natural Language Processing due to their ability to scale across multiple tasks rapidly by pre-training a single model, followed by task-specific fine-tuning. Furthermore, multilingual versions of such models like XLM-R and mBERT have given promising results in zero-shot cross-lingual transfer, potentially enabling NLP applications in many under-served and under-resourced languages. Due to this initial success, pre-trained models are being used as 'Universal Language Models' as the starting point across diverse tasks, domains, and languages. This work explores the notion of 'Universality' by identifying seven dimensions across which a universal model should be able to scale, that is, perform equally well or reasonably well, to be useful across diverse settings. We outline the current theoretical and empirical results that support model performance across these dimensions, along with extensions that may help address some of their current limitations. Through this survey, we lay the foundation for understanding the capabilities and limitations of massive contextual language models and help discern research gaps and directions for future work to make these LMs inclusive and fair to diverse applications, users, and linguistic phenomena.

1 Introduction

Language Models (LMs) have evolved considerably in the past decade, starting from the introduction of Word2Vec (Mikolov et al., 2013) to the more recent transformer-based deep models like BERT (Devlin et al., 2019) and its successors. When fine-tuned with task-specific data, pre-trained LMs can be adapted to several different settings, i.e., tasks, domains, and even languages, as these LMs have been extended to multiple languages in the multilingual versions like m-BERT and derivatives. These models can be thought of as 'Universal' because of their potential to be utilized 'universally' in several different application scenarios.[1]

The merits of transfer learning or pre-training word representations have been known for a long time. Moreover, the recent advancements in large-scale deep learning have pushed the boundaries of intensive computation and tremendous amounts of data that can be used to pre-train LMs. However, pre-training is resource-intensive and is not carried out for specific scenarios. Instead, massive LMs are deployed into downstream applications with potentially billions of users around the world. This makes 'Universality' a vital characteristic as the models must be inclusive towards a variety of language usage.

The key contributions of this paper are:

- We formally define 'Universality' by selecting seven dimensions- language, multilingualism, task, domain, medium of expression, geography and demography, and time period - that capture a variety of language usage.
- We curate the current empirical and theoretical results that provide evidence of scaling LMs across these dimensions and identify the capabilities and gaps in these models.
- We outline extensions to these models that can help in overcoming current limitations to become truly universal, thus serving a larger number of end-users and scenarios.

[*]Work done while at Microsoft Research, India

[1]Throughout the rest of the paper – "these models", "LMs", "general domain LMs", "contextual LMs", "universal LMs" and all such terms refers to models including but not limited to ELMo, BERT, RoBERTa, GPT their variants, successors and multilingual versions

2 Universality and its Dimensions

Universality can mean many things, and the associated philosophical debate is beyond the scope of this study. Our work aims not to provide a complete or exhaustive list of capabilities expected out of the model but a list of aspects that can be considered as a starting goal to achieve universality.

Our definition of universality spans seven dimensions. These are: language, multilingualism, task, domain, medium of expression, demography and geography, and time period. We selected these dimensions to cover a broad spectrum of language usage and a diverse set of NLP applications. Ideally, a truly universal model should perform strongly, or at least reasonably, across them. We present a detailed analysis of the capabilities and limitations of LMs in these dimensions in the subsequent sections. This is followed by extensions, which are techniques that can be leveraged to overcome the limitations in the particular dimension.

2.1 Reasoning for Selection of Dimensions

Firstly, it is important to re-iterate that this list of seven dimensions does not intend to be an exhaustive one. Rather, these dimensions have been selected so as to cover a broad spectrum of language usage. The reasons why each of these is important for a general-purpose LM are:

Language Massive multilingual models that can support close to 100 languages at a time are quickly becoming standard for building language technologies that cater to a wide and linguistically diverse population. However, while high-resources languages are well served, many low-resource languages are left behind (Joshi et al., 2021). Thus it is important to understand where large scale multilingual LMs stands in terms of availability, evaluation, and performance along the dimension of Language.

Multilingualism LMs are increasingly being deployed into user-facing applications and thus need to deal with real-world language usage. In multilingual (or bilingual) communities, usage of multiple languages at once gives rise to many language variations such as code-mixing, that the model will need to process. For ascertaining how well models can deal with these linguistic phenomena, understanding the capabilities and limitations of the models along this dimension becomes important.

Task LMs are increasingly becoming the standard component of most NLP pipelines. As a result, it is important to study how well they adapt to various different tasks for which they are used.

Domain Typically, LMs are trained on general purpose language, such as that obtained from Wikipedia or the Web. As such, their training data has limited signals for complex vocabulary that is common for a specialized domain such as medical, financial, legal, etc. However, real-world applications of LMs may require it to deal with information from different domains. Thus it is important to understand the limitations of these models when employed in different domain settings.

Medium of Expression Whether the language being processed is from a formal email or from an informal utterance on social media, can make significant difference in its syntactic and semantic properties. LMs being deployed in applications that span across different media are thus bound to come across linguistic variations induced due to the medium of expression. This makes it important to understand how LMs perform across different mediums of expression.

Geography and Demography Most languages in the world, including English, have multiple dialects that are influenced by geographic and demographic factors. The applications that are developed using LMs are intended to be used across the world, spanning users belonging to varied demography. It is hence important that the LMs are inclusive towards different forms of language usage and not just cater to a 'standardized' dialect of the language. Hence, it is important to understand how LMs perform across regional and social language dialects.

Time Period Given the high financial and environmental costs of training language models, a single model can be anticipated to be used for long periods of time. Language, however, changes extremely rapidly. Events happening around the world cause constant changes in the vocabulary and semantics of a language. Thus, it is important for LMs to be robust towards new word senses, sentence structures, etc. It is thus necessary to evaluate models on the dimension of Time Period as they are bound to come across language belonging to different points in the history.

The following sections go over each of these

Table 1: Languages and Tasks covered by different datasets and benchmarks

Datasets	Langs.	Tasks
WikiANN	176[2]	NER
UD v2	90	POS, dependencies
XNLI	15	NLI
XQuAD	11	QA
MLDoc	8	Document classification
MLQA	7	QA
PAWS-X	6	Paraphrase identification
Benchmarks[3]		
XTREME	40	NLI, POS, NER, QA, Paraphrase identification, Sentence retrieval
XGLUE	19	NER, POS, QA, NLI, News classification, QA matching, Paraphrase identification, Query-ad matching, Web page ranking, Question generation, News title generation

dimensions to describe the limitations and capabilities of models along each of them.

3 Language

There are over 7000 languages in the world. There is an increased demand for multilingual systems as information technologies penetrate more lives globally. The largest available LMs include mBERT (Devlin et al., 2019), XLM-R (Conneau et al., 2020a), and mT5 (Xue et al., 2020) serve 104, 100, and 101 languages respectively. It is clear that they are far from universal in terms of language coverage compared to the number of languages in the world. Further, there is an expectation that massive multilingual LMs will perform equally, or at least reasonably, well on all the languages they serve.

The limited availability of evaluation benchmarks is a major bottleneck in knowing how LMs perform across the languages they are pre-trained on. Table 1 shows that the largest benchmark, XTREME (Hu et al., 2020), covers less than half of the total number of languages that these LMs are trained on. Moreover, other than datasets for syntactic tasks like NER (Rahimi et al., 2019), and POS (Nivre et al., 2016), the largest available semantic task dataset, XNLI (Conneau et al., 2018), covers only 15 languages. Although LMs are tested on individual tasks or languages that may not be covered in these benchmarks, overall, there is a considerable reliance on standard benchmarks to make modelling choices. Thus, how well LMs perform in the untested languages remains unanswered.

There is a great disparity in performance across the languages that are tested through these benchmarks. A general observation is that the performance of low resource languages continues to be lower than high resource languages. The extent to which cross-lingual transfer helps improve performance varies across languages. Studies that empirically support these claims are:

Cross-script transfer is not equally good across languages in mBERT. Ahmad et al. 2019a find that cross-script cross-lingual transfer is effective in the case of Hindi and Urdu, whereas this is not observed between English and Japanese.

Word order differences across language leads to worse cross-lingual transfer (Ahmad et al., 2019a). The correlation between word ordering distance and cross-lingual transfer is found to be high in the experiments by Ahmad et al. 2019b. K et al. 2020 also find that word order has a significant bearing on transfer.

Contrary to common intuition, K et al. 2020 find that shared vocabulary does not affect Universality or generalization across languages considerably. Artetxe et al. 2020a also find that 'effective vocabulary size per language' affects cross-lingual performance rather than joint or disjoint vocabulary of multiple languages.

In massively multilingual LMs, where typically a joint vocabulary across languages is used, languages tend to compete for the allocations in the shared vocabulary. Siddhant et al. 2020 show that increasing number of languages may worsen performance compared to models with fewer languages. This is similar to the findings of Wu and Dredze 2019. Thus, limiting pre-training to only the required languages needed for the downstream tasks may be more beneficial. Conneau et al. 2020a coined the term *"curse of multilinguality"* for this phenomenon and pointed to the trade-off between model performance and language coverage. This result is also shown in MuRIL, a BERT model trained on 17 Indian languages, which outperforms mBERT on the XTREME benchmark significantly across all languages (MuRIL, 2021). Similarly, clustering languages and using different multilingual model for each group, rather than one massive model, gives better performance in Neural Machine Translation (Tan et al., 2019).

Wu and Dredze 2019 observe that mBERT does

[2] Balanced version

[3] Each task may cover only a subset of languages

not transfer well between distant languages. Further, they conclude that while mBERT may perform very well in cross-lingual transfer compared to other models, it still falls short of models that have been trained with cross-lingual signals like bitext, bilingual dictionaries, or limited target language supervision.

Answering whether all languages in mBERT are equally well represented, Wu and Dredze 2020 find that mBERT does not learn high-quality representations for all languages, especially for low resource languages. The bottom 30% languages in terms of data size perform even worse than a non-BERT model for NER. For low resource languages, the combined effect of less data for pre-training and annotated data for fine-tuning compounds together leading to worsening of their performance. On the other end of the spectrum, the top 10% languages are hurt by joint training as mBERT performs worse than monolingual baselines of NER.

To summarize, universality in the language dimension has three levels. At the highest level, the largest models available today span only around 100 of the 7000+ languages globally and thus are far from universal in terms of language coverage. Secondly, out of the languages that these models are trained on, not all of them are evaluated, implying that we do not have enough information to make generalized claims of universality in performance for the languages that the LMs support. Finally, at the lowest level, the performance is not uniform across the tested languages. The performance of lower resourced languages is lower than that of higher-resourced languages. Increasing the number of languages hurts performance at both ends of the spectrum, and cross-lingual transfer is non-uniform and dependent on many factors.

Extensions: Monolingual models learn generalizable representations and can be adapted to new languages without joint training. Conneau et al. 2020b show that the representations learned by monolingual models without any shared vocabulary align with each other and can be adapted to a new language. Similarly, Artetxe et al. 2020b study the transfer of monolingual representations to new languages without using shared vocabulary or joint training. They propose a zero-shot cross-lingual transfer technique where the resultant model is a monolingual LM adapted to a new language. Tela et al. 2020 study adaptation to the extremely low resourced language, Tigrinya. They find that English XLNet generalizes better than BERT and mBERT, which is surprising given that mBERT is trained in multiple languages. Thus, the adaptation of monolingual models may help in extending LMs to new low-resource languages.

Wang et al. 2020 enlarge mBERT's vocabulary and continue pre-training on 27 target languages, out of which 11 are new. They observe performance improvement in zero-shot cross-lingual NER for all 27 languages. The extension benefits both the existing and newly added languages. The drawback is that the base model (mBERT) is biased towards the target languages, downgrading performance on non-target languages.

The data and compute cost of training LMs from scratch poses a major limitation, especially for low-resourced languages. Tran 2020 propose a data and compute efficient technique to circumvent the need of training language-specific models from scratch. They learn target language word-embeddings from an English LM while keeping the pre-trained encoder layer fixed. The English and target language LMs are then both fine-tuned to obtain a bilingual LM. This technique performs better than mBERT and XLM-R on XNLI in 5 of 6 languages with different amounts of resources.

Chi et al. 2020 combine the cross-lingual transfer of a multilingual LM with a task-specific monolingual LM to improve zero-shot cross-lingual classification. The source-language monolingual 'teacher' model provides supervision for the downstream task, and the multilingual model acts as a 'student'. The method outperforms direct multilingual fine-tuning for zero-shot cross-lingual sentiment analysis and XNLI in most of the languages.

Pfeiffer et al. 2020 propose an adaptor based modular framework that mitigates the curse of multilinguality and adapts a multilingual model to arbitrary tasks and languages using language and task-specific adaptors. Their method gives state-of-the-art results for cross-lingual transfer among typologically diverse languages across tasks including NER, causal commonsense reasoning, and QA.

4 Multilingualism

In multilingual communities, several linguistic phenomena lead to variation in language usage. While some of these are well-known and studied, others do not get enough attention. Universal LMs should be able to deal with these phenomena as we can expect them to encounter such forms of language when deployed in user-facing applications.

Code-mixing, or using two more languages in a single utterance, is common in multilingual communities. LMs may not perform optimally in the presence of such code-mixing. Multilingual models like mBERT are not pre-trained with mixed language data, which leaves the model under-prepared for code-switched settings resulting in sub-optimal performance (Khanuja et al., 2020). This can be overcome to a certain extent by training using other data, such as social media, but it is unlikely to cover all the forms of code-switching produced by multilinguals.

Romanization of languages to the Latin script has increased with the advent of digitization of communication. While some models like XLM-R (Conneau et al., 2020a) and MuRIL (MuRIL, 2021) use romanized versions of some languages in training, the effectiveness of these LMs on Romanized (or more generally transliterated) text is still unclear.

Diglossia is a kind of multilingualism where a single community uses a substantially different language dialect in different communication settings (Ferguson, 1959). Since data from the internet is used in training, it is likely that LMs cannot handle diglossia. However, there are no studies that concretely prove (or disprove) this.

To summarize, apart from code-mixing, there has been very little work in recognizing, understanding, or improving LMs for different phenomena arising due to multilingualism, making the dimension under-represented in the study of LMs.

Extensions: Efforts have been made to improve LMs, particularly mBERT to deal with code mixed data. Khanuja et al. 2020 present a modified version of mBERT which performs better than standard mBERT in English-Hindi and English-Spanish code mixed data using synthetically generated code-mixed data for continued pre-training.

5 Task

NLP applications range from syntactic tasks, like POS, NER, etc. to semantic tasks like NLI, QA, etc. LMs learn task-agnostic representations and can be fine-tuned with task-specific data or used in task-specific architectures as features. Thus, Universal LMs should adapt well to a wide variety of tasks.

Like languages, the extent of evaluation on different NLP tasks is constrained by the availability of benchmarks that span various tasks. As shown in Table 1, only a small fraction of the large number of tasks studied in NLP are evaluated by benchmarks. While there are many other task-specific datasets, the success of LMs is associated with performance on these benchmarks rather than a wider variety of tasks.

Universal Language Model Fine-tuning for Text classification (ULMFiT) uses discriminative fine-tuning, gradual unfreezing of layers, and slanted triangular learning rates for target-specific fine-tuning, giving better performance on multiple tasks (Howard and Ruder, 2018). This work also explicitly defines the term ‘universal’, in their context as referring to – applicable to all tasks in text classification, using a single training process and architecture, usable without feature-engineering, and not requiring additional in-domain data.

Masked language modeling (MLM) is the most generalizable pre-training objective for the extent of transfer among twelve pre-training objectives for nine target tasks (Liu et al., 2019).

Data size is important in effective pre-training of LMs (Liu et al., 2019) but transfer gains between source and target tasks are also possible with smaller source datasets (Vu et al., 2020b).

Similarity between source and target tasks is important for transfer gains. Liu et al. 2019 find that closeness in pre-training objective and target task is important for transfer. Peters et al. 2019 find that while feature extraction and fine-tuning of LMs give similar performance, exceptions occur when the source and target tasks are either very similar or very dissimilar. Vu et al. 2020b also find that similarity is important, especially in low-resource scenarios, but, exceptions of transfer gains between dissimilar tasks are possible.

To summarize, LMs are universal in the task dimension owing to task-specific architectures or fine-tuning. However, the success of LMs is often associated with few benchmarks which cover limited tasks. The similarity between tasks, data size, and pre-training objectives are keys to transfer gains, which are important for Universality.

Extensions: Pattern Exploiting Training (PET) (Schick and Schütze, 2020a) reformulates tasks as cloze questions,[4] making them the same as the MLM objective, requiring less data with no additional fine-tuning or a task-specific architecture to achieve remarkable zero-shot and few-shot perfor-

[4]Cloze questions are statements with exactly one masked token.

mance. Using PET with ALBERT few-shot performance competitive to GPT-3, which is 780 times larger in terms of the number of parameters compared to ALBERT, is obtained (Schick and Schütze, 2020b). The recently introduced T5 model (Raffel et al., 2020) leverages a text-to-text framework to enabling a single architecture to perform multiple tasks and achieving state-of-art results. These are concrete steps to enable universality towards tasks, as a single model can be built to generalize across solving multiple tasks.

6 Domain

NLP models are applied to many real-world applications in different domains like medical, scientific, legal, financial etc. A universal model in this dimension should adapt to different domains or scenarios without loss of performance.

Processing domain-specific language often requires the processing of specialized vocabulary and language usage. Even though LMs learn some implicit clusters of domains (Aharoni and Goldberg, 2020), this may not be enough and specialized domain-specific LMs are needed (Lee et al., 2020; Chalkidis et al., 2020; Beltagy et al., 2019; Huang et al., 2019; Araci, 2019; Gu et al., 2020).

Despite the success of general domain LMs, it is found that pre-training LM on in-domain data improves performance across high and low resource settings (Gururangan et al., 2020).

Lee et al. 2020 introduced BioBERT, learned using continued pre-training of BERT on medical text, which performed better than BERT on biomedical tasks. In contrast, Gu et al. 2020 introduce PubMedBERT and challenge the benefits of out-of-domain data in pre-training by showing that pre-training LM from scratch on in-domain data (if available) is better than mixed or continual pre-training. Huang et al. 2019 propose the clinicalBERT model that is pre-trained on clinical notes text corpora, which learns better relationships among medical concepts and outperforms general domain LMs in clinical tasks.

Chalkidis et al. 2020 introduce Legal-BERT and observe that continual pre-training of BERT or training it from scratch with legal data both perform similarly and significantly better than using BERT off the shelf. Beltagy et al. 2019 release SciBERT, trained from scratch on scientific publications giving better performance on scientific tasks. Araci 2019 use domain adaptation and transfer learning to develop FinBERT, achieving state-of-the-art performance in the financial domain.

Performance of domain-specific LM can degrade on general domain tasks (Gu et al., 2020; Xu et al., 2020; Thompson et al., 2019; Rongali et al., 2020). This phenomenon is also known as catastrophic forgetting and prevents the LM from being truly universal.

To summarize, LMs are not universal in the domain dimension, and different domain-specific LMs have been introduced to cater to this requirement. Domain-specific LMs are either trained from scratch or by mixed or continual pre-training of existing LMs. While none of these techniques are clear winners, performance degradation in general domain tasks is observed in many cases.

Extensions: Vu et al. 2020a study adversarial masking strategies to learn specific target domain vocabulary along with continual pre-training by carefully selecting tokens to be masked, leading to better domain adaptation performance across multiple source and target domains.

MuTSPad (Multi-Task Supervised Pretraining and Adaptation) (Meftah et al., 2020) leverages hierarchical learning of a multi-task model on high-resource domain followed by fine-tuning on multiple tasks on the low-resource target domain.

Ben-David et al. 2020 extend the pivot-based transfer learning to transformer-based LMs by developing PERL (Pivot-based Encoder Representation of Language), that uses continual pre-training with MLM to learn representations that reduce the gap between source and target domains followed by fine-tuning for the downstream classification task. The pivot is selected such that the source and target domain labels have greater mutual information to facilitate a good transfer.

Jiang and Zhai 2007 propose several heuristics like removing misleading examples from the source domain, assigning more weight to target domain instances, and augmenting target training instances with predicted labels for better domain adaptation from a distributional perspective.

Various methods that are computationally efficient (Poerner et al., 2020), use more effective adversarial training (Ma et al., 2019), and reduce the requirement of annotated data in low-resource settings (Hazen et al., 2019) have been proposed for computation and data efficient domain adaptation.

Rongali et al. 2020 overcome catastrophic forgetting, out-performing domain-specific LMs while maintaining performance on general domain tasks.

7 Medium of Expression

Language varies with the medium of expression. There are syntactic and semantic alterations like the use of ill-formed sentence or grammatical structure, inflections, slang, compressions, and abbreviations due to limited space, familiarity with the audience, and communicative intent. Universal LMs should be robust to such variations. Often these specialized settings are simply treated as a different domain (Qudar and Mago, 2020; Nguyen et al., 2020). However, in this work, we treat 'domain' as specialized fields of expertise. The current discussion pertains to the medium in which language is used.

Language in social media and texting may not follow conventions of written language. Sentences are often shorter or grammatically ill-formed and may not be coherent enough for LMs that rely on contextual information (Eisenstein, 2013; Han et al., 2013; Choudhury et al., 2007).

LMs perform sub-optimally on non-standard language as compared to specialized LMs. BERTweet (Nguyen et al., 2020), a Roberta-based LM trained on English tweets, outperforms both RoBERTa and XLM-R base models even though RoBERTa and XLM-R use 2 times and 3.75 times more data, respectively. TweetBERT (Qudar and Mago, 2020), another LM trained on tweets outperforms seven general domain LMs.

To summarize, the language used in different media of expression is substantially different. The limited amount of investigation in this direction indicates that LMs are not universal to these variations and perform sub-optimally on the language used in social media and texting.

Extensions: Dai et al. (2020) propose cost-effective training by appropriate selection of additional data for training a LM from a Twitter corpus.

8 Geography and Demography

Standard and non-standard dialects, both social and regional, lead to varied word, and language use (Labov, 1980; Milroy, 1992; Tagliamonte, 2006; Wolfram and Schilling, 2015; Nguyen et al., 2016). Regional dialect refers to the varied usage of the same language across different places. For example, the use of 'wicked' can refer to bad or evil ("he is a wicked man"), or as an intensifier to adverbs ("my son is wicked smart") (Bamman et al., 2014a; Kulkarni et al., 2016). Sociolects, or social dialects, similar to regional dialects, are language dialects dependant on social variables like age, race, gender, socio-economic status, ethnicity, etc. (Nguyen et al., 2016). Geographic and demographic variations stem from grammatical, phonological, syntactic, lexical, semantic features, or any combinations, making it difficult to capture and evaluate.

Since LMs are trained on standard language dialects, non-standard dialects spoken by millions of people are largely ignored. Such a system can result in bias against specific cultural or geographical communities in user-facing applications leading to ethical implications in building fair NLP systems.

A Universal LM should be sensitive to semantic shifts arising from its users' demographic or geographical diversity. Taking these variations into consideration has resulted in improved performance and personalization of applications like conversational agents, sentiment analysis, word prediction, cyber-bullying detection, and machine translation (Östling and Tiedemann, 2017; Rahimi et al., 2017; Mirkin and Meunier, 2015; Hovy, 2015; Hovy and Søgaard, 2015; Stoop and van den Bosch, 2014; Volkova et al., 2013).

Kulkarni et al. 2016 present a novel approach to quantify semantic shift that is statistically significant across geographical regions and propose a new measure of dialect similarity to establish how close the language in two regions is. Demszky et al. 2020 focus on Indian English and show that dialect features can be learned given very limited data with strong performance.

In the 2020 VarDial evaluation task for Romanian Dialect Identification, an SVM ensemble based on word and character n-grams outperformed fine-tuned Romanian BERT model. These results are consistent with the earlier evaluation results of VarDial where shallow models outperformed deep models. In Social Media Variety Geolocation, predicting geolocation (coordinates) from text, the best performance was obtained by a BERT architecture with a double regression classification output. In contrast, the next two best models were both shallow. (Gaman et al., 2020).

Demographic features like age, gender have been predicted from language usage Peersman et al. (2011); Morgan-Lopez et al. (2017), whereas social class or ethnicity receive less attention (Mohammady Ardehaly and Culotta, 2015). Prediction of demographic features from language use quantifies the correlation between social variables and social dialects. While individuals may intentionally or naturally digress from such conventions, these statistical patterns are a cornerstone for studying the interaction between society and language in computational sociolinguistics (Nguyen et al., 2016).

There are some worrisome findings of bias of performance across age, ethnicity, or gender in contextual LMs. (Hovy and Søgaard, 2015) find that a POS tagger trained on the English Penn treebank performed better on texts written by older authors. Tan et al. 2020 show bias against non-standard English (in this case Singaporean English) in BERT. Bhardwaj et al. 2020 expose gender bias in BERT by showing that the model assigns lower (or higher) scores consistently for sentences that contain words indicating gender in cases where gender information should have no bearing.

To summarize, the influence of geography and demography on language usage is well known, and LMs must be sensitive and inclusive of such variation. However, there has been limited, albeit now growing, attention to these factors. In some cases, shallow models have outperformed deep models in recognizing semantic shifts, and there is evidence of bias against particular social groups.

Extensions: Bamman et al. 2014b learn language representations that take geographical situations or variations into account by enriching Vector space word representations (word2vec) with geographical knowledge from metadata about authors. Hovy and Purschke 2018 employ retrofitting for including geographic information to capture regional variation in continuous regional distribution and at a fine-grained level using online posts in German and the corresponding cities of their authors as labels to create document embeddings. While these techniques do not involve any contextual LMs, such representations and retrofitting can be extended to contextual LMs.

Tan et al. 2020 propose an adversarial approach to make models like BERT more robust to non-standard forms of English using inflectional morphology perturbations.

Debiasing techniques such as the ones studied in (Bolukbasi et al., 2016; Kaneko and Bollegala, 2019) can remove gender stereotypes from pre-trained word embeddings.

9 Time Period

Language evolves continuously, and individual word meanings can change significantly over the years (Cook et al., 2014; Kim et al., 2014; Hamilton et al., 2016). Most of the data used in LM pre-training is from the late 20th century. Thus, whether these models can handle word senses across timescales is a pertinent question. Universal LMs should appropriately deal with language with a nuanced understanding of diachronic semantic change (DSC) because, when deployed in downstream applications, such variations may be encountered and misinterpreted. Some of the studies we mention below are not strictly focused on contextual LMs. Nevertheless, we find it important to note such research as we hope it can be extended to contextual LMs in the future.

The intensity of the change of meaning of different words is different – some are more subtly changed than the others. Kim et al. 2014 trace a period from 1900-2009, obtain year-specific word embeddings on the Google Books N-grams corpus, and pinpoint the extent and time-period of occurrence of semantic shift.

Hamilton et al. 2016 evaluate static word embeddings for known historical changes using corpora spanning four languages and two historical periods. They create diachronic embeddings by learning separate representations across the time periods followed by alignment over different time scales.

DSC can be detected by clustering word sensed. Mitra et al. 2014 organize words in a time-period specific graph where its nearest neighbors are co-occurring words, and word-senses are clustered. The shift in word sense or the emergence of a new word sense can be identified by the change of cluster for a particular word. Giulianelli et al. 2020 perform clustering over usage types in BERT and use the contextual property of LM to quantify semantic change instead of relying on a specific set of word senses.

DSC evaluation lacks standardization. Schlechtweg et al. 2019 perform a large scale evaluation on German, revealing the best set of parameters for optimal performance, compare various state-of-the-art methods, and outline improvements for better performance.

Shallow models can outperform contextual LMs in identifying semantic shift. Schlechtweg et al. 2019 show that a shallow skip-gram model with negative sampling, orthogonal alignment, and cosine distance performs best in identifying DSC in German. Kaiser et al. 2020 reconfirm this by using a similar model to obtain the first position in the DIACR-Ita shared task (Basile et al., 2020) on Italian DSC. Similar findings of only limited success in contextual LMs are reported in the shared task on Unsupervised Lexical Semantic Change Detection in 5 languages hosted at SemEval 2020 (Schlechtweg et al., 2020).

While the success in identifying these shifts may be limited, (Rodina et al., 2020) find that DSC identified by contextual LMs can have a strong correlation with human judgment of change.

Within contextual LMs, BERT and ELMo perform similarly for Russian. Rodina et al. 2020 show that neither BERT nor Elmo outperform each other when fine-tuned using historical text in Russian to detect semantic change. Moreover, results from the shared task on Unsupervised Lexical Semantic Change Detection in 5 languages hosted at SemEval 2020 (Schlechtweg et al., 2020) show that systems performing well over one language may not perform as well for other languages.

To summarize, time period is under-studied and there is little understanding of whether contextual LMs can handle such nuanced language variation. For the closely related task of DSC, shallow models can outperform deep LMs, and performance can vary greatly across languages.

Extensions: Rudolph and Blei 2017 develop dynamic word-embeddings with an attribute of time that captures the semantic shifts in word meanings in sequential historical data on top of Bernoulli embeddings such that representations are shared within specific time periods rather than throughout the corpus. Similarly, Bamler and Mandt 2017 use timestamped data to build static probabilistic representation for tracing semantic change.

To mitigate the problem of using different representations of words over different time periods, Hu et al. 2019 propose a framework for tracking and representing word senses by leveraging pre-trained BERT embeddings and Oxford dictionary data to learn fine-grained senses.

10 Conclusion

Deep Contextual LMs are being applied today to various different applications due to their perceived 'Universality'. In this work, we attempt to holistically define 'Universality' to encompass a wide variety of scenarios and linguistic phenomena.

We define Universality using seven dimensions: Language, Multilingualism, Task, Domain, Medium of Expression, Geography and Demography, and Time Period. These dimensions result in unique variations in language usage that are commonly encountered in real-life scenarios. We aim for this definition to be sound rather than complete. That is, a model should strive to achieve Universality in these dimensions, but they are in no way a complete, exhaustive list of everything the model needs to be capable of.

We survey research across all the dimensions and find that: First, while dimensions like language, task, and domain are more widely studied, other dimensions, especially multilingualism, geography and demography, and time period receive less attention. Second, limited evaluation benchmarks constrain the complete understanding of capabilities even in the more studied dimensions. Third, language variation arising in specific scenarios of demography, geography, time period, multilingualism, and medium of expression is often studied in an isolated manner.

The dimensions we survey are a starting point that LMs can aim to be inclusive towards in order to serve a diverse set of users and scenarios. Large contextual LMs may not be the optimal choice for all scenarios, with shallow, task-specific models sometimes leading to better outcomes. Overall, 'Universality' is yet to be fully understood, studied, and achieved. We hope that this work will lay the foundation to understanding the capabilities and limitations of LMs and spur further research into making models more inclusive and fair.

References

Roee Aharoni and Yoav Goldberg. 2020. Unsupervised domain clusters in pretrained language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7747–7763, Online. Association for Computational Linguistics.

Wasi Ahmad, Zhisong Zhang, Xuezhe Ma, Eduard Hovy, Kai-Wei Chang, and Nanyun Peng. 2019a. On difficulties of cross-lingual transfer with order differences: A case study on dependency parsing. 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 2440–2452, Minneapolis, Minnesota. Association for Computational Linguistics.

Wasi Ahmad, Zhisong Zhang, Xuezhe Ma, Eduard Hovy, Kai-Wei Chang, and Nanyun Peng. 2019b. On difficulties of cross-lingual transfer with order differences: A case study on dependency parsing. 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 2440–2452, Minneapolis, Minnesota. Association for Computational Linguistics.

Dogu Araci. 2019. Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020a. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online. Association for Computational Linguistics.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020b. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online. Association for Computational Linguistics.

Robert Bamler and Stephan Mandt. 2017. Dynamic word embeddings. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 380–389. PMLR.

David Bamman, Chris Dyer, and Noah A. Smith. 2014a. Distributed representations of geographically situated language. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 828–834, Baltimore, Maryland. Association for Computational Linguistics.

David Bamman, Chris Dyer, and Noah A. Smith. 2014b. Distributed representations of geographically situated language. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 828–834, Baltimore, Maryland. Association for Computational Linguistics.

Pierpaolo Basile, Annalina Caputo, Tommaso Caselli, Pierluigi Cassotti, and Rossella Varvara. 2020. Diacr-ita@ evalita2020: Overview of the evalita2020 diachronic lexical semantics (diacr-ita) task. *Proceedings of the 7th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA 2020), Online. CEUR. org*.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. 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 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Eyal Ben-David, Carmel Rabinovitz, and Roi Reichart. 2020. PERL: Pivot-based domain adaptation for pre-trained deep contextualized embedding models. *Transactions of the Association for Computational Linguistics*, 8:504–521.

Rishabh Bhardwaj, Navonil Majumder, and Soujanya Poria. 2020. Investigating gender bias in bert.

Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam Tauman Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4349–4357.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. LEGAL-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, Online. Association for Computational Linguistics.

Zewen Chi, Li Dong, Furu Wei, Xianling Mao, and Heyan Huang. 2020. Can monolingual pretrained models help cross-lingual classification? 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 12–17, Suzhou, China. Association for Computational Linguistics.

Monojit Choudhury, Rahul Saraf, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar, and Anupam Basu. 2007. Investigation and modeling of the structure of texting language. *International Journal of Document Analysis and Recognition (IJDAR)*, 10(3-4):157–174.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020a. 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, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium. Association for Computational Linguistics.

Alexis Conneau, Shijie Wu, Haoran Li, Luke Zettlemoyer, and Veselin Stoyanov. 2020b. Emerging cross-lingual structure in pretrained language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6022–6034, Online. Association for Computational Linguistics.

Paul Cook, Jey Han Lau, Diana McCarthy, and Timothy Baldwin. 2014. Novel word-sense identification. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1624–1635, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Xiang Dai, Sarvnaz Karimi, Ben Hachey, and Cecile Paris. 2020. Cost-effective selection of pretraining data: A case study of pretraining BERT on social media. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1675–1681, Online. Association for Computational Linguistics.

Dorottya Demszky, Devyani Sharma, Jonathan H Clark, Vinodkumar Prabhakaran, and Jacob Eisenstein. 2020. Learning to recognize dialect features. *arXiv preprint arXiv:2010.12707*.

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.

Jacob Eisenstein. 2013. What to do about bad language on the internet. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 359–369, Atlanta, Georgia. Association for Computational Linguistics.

Charles A Ferguson. 1959. Diglossia. *word*, 15(2):325–340.

Mihaela Gaman, Dirk Hovy, Radu Tudor Ionescu, Heidi Jauhiainen, Tommi Jauhiainen, Krister Lindén, Nikola Ljubešić, Niko Partanen, Christoph Purschke, Yves Scherrer, and Marcos Zampieri. 2020. A report on the VarDial evaluation campaign 2020. In *Proceedings of the 7th Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 1–14, Barcelona, Spain (Online). International Committee on Computational Linguistics (ICCL).

Mario Giulianelli, Marco Del Tredici, and Raquel Fernández. 2020. Analysing lexical semantic change with contextualised word representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3960–3973, Online. Association for Computational Linguistics.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2020. Domain-specific language model pretraining for biomedical natural language processing. *arXiv preprint arXiv:2007.15779*.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1489–1501, Berlin, Germany. Association for Computational Linguistics.

Bo Han, Paul Cook, and Timothy Baldwin. 2013. Lexical normalization for social media text. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(1):1–27.

Timothy J Hazen, Shehzaad Dhuliawala, and Daniel Boies. 2019. Towards domain adaptation from limited data for question answering using deep neural networks. *arXiv preprint arXiv:1911.02655*.

Dirk Hovy. 2015. Demographic factors improve classification performance. 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 752–762, Beijing, China. Association for Computational Linguistics.

Dirk Hovy and Christoph Purschke. 2018. Capturing regional variation with distributed place representations and geographic retrofitting. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4383–4394, Brussels, Belgium. Association for Computational Linguistics.

Dirk Hovy and Anders Søgaard. 2015. Tagging performance correlates with author age. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 483–488, Beijing, China. Association for Computational Linguistics.

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 (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. *International Conference on Machine Learning*, pages 4411–4421.

Renfen Hu, Shen Li, and Shichen Liang. 2019. Diachronic sense modeling with deep contextualized word embeddings: An ecological view. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3899–3908, Florence, Italy. Association for Computational Linguistics.

Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. 2019. Clinicalbert: Modeling clinical notes and predicting hospital readmission. *arXiv preprint arXiv:1904.05342*.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in NLP. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 264–271, Prague, Czech Republic. Association for Computational Linguistics.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2021. The state and fate of linguistic diversity and inclusion in the nlp world.

Karthikeyan K, Zihan Wang, Stephen Mayhew, and Dan Roth. 2020. Cross-lingual ability of multilingual BERT: an empirical study. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Jens Kaiser, Dominik Schlechtweg, and Sabine Schulte im Walde. 2020. Op-ims@ diacr-ita: Back to the roots: Sgns+ op+ cd still rocks semantic change detection. *arXiv preprint arXiv:2011.03258*.

Masahiro Kaneko and Danushka Bollegala. 2019. Gender-preserving debiasing for pre-trained word embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1641–1650, Florence, Italy. Association for Computational Linguistics.

Simran Khanuja, Sandipan Dandapat, Anirudh Srinivasan, Sunayana Sitaram, and Monojit Choudhury. 2020. GLUECoS: An evaluation benchmark for code-switched NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3575–3585, Online. Association for Computational Linguistics.

Yoon Kim, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. 2014. Temporal analysis of language through neural language models. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 61–65, Baltimore, MD, USA. Association for Computational Linguistics.

Vivek Kulkarni, Bryan Perozzi, Steven Skiena, et al. 2016. Freshman or fresher? quantifying the geographic variation of language in online social media. In *ICWSM*, pages 615–618.

William Labov. 1980. *Locating language in time and space*. Academic Press New York.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual 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 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Xiaofei Ma, Peng Xu, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2019. Domain adaptation with BERT-based domain classification and data selection. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 76–83, Hong Kong, China. Association for Computational Linguistics.

Sara Meftah, Nasredine Semmar, Mohamed-Ayoub Tahiri, Youssef Tamaazousti, Hassane Essafi, and Fatiha Sadat. 2020. Multi-task supervised pretraining for neural domain adaptation. In *Proceedings of the Eighth International Workshop on Natural Language Processing for Social Media*, pages 61–71, Online. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

James Milroy. 1992. *Linguistic variation and change: On the historical sociolinguistics of English*. B. Blackwell.

Shachar Mirkin and Jean-Luc Meunier. 2015. Personalized machine translation: Predicting translational preferences. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2019–2025, Lisbon, Portugal. Association for Computational Linguistics.

Sunny Mitra, Ritwik Mitra, Martin Riedl, Chris Biemann, Animesh Mukherjee, and Pawan Goyal. 2014. That's sick dude!: Automatic identification of word sense change across different timescales. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1020–1029, Baltimore, Maryland. Association for Computational Linguistics.

Ehsan Mohammady Ardehaly and Aron Culotta. 2015. Inferring latent attributes of Twitter users with label regularization. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 185–195, Denver, Colorado. Association for Computational Linguistics.

Antonio A Morgan-Lopez, Annice E Kim, Robert F Chew, and Paul Ruddle. 2017. Predicting age groups of twitter users based on language and metadata features. *PloS one*, 12(8):e0183537.

MuRIL. 2021. Multilingual Representations for Indian Languages. https://tfhub.dev/google/MuRIL/1. Accessed: 2021-01-29.

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. BERTweet: A pre-trained language model for English tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14, Online. Association for Computational Linguistics.

Dong Nguyen, A Seza Doğruöz, Carolyn P Rosé, and Franciska de Jong. 2016. Computational sociolinguistics: A survey. *Computational linguistics*, 42(3):537–593.

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'16)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).

Robert Östling and Jörg Tiedemann. 2017. Continuous multilinguality with language vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 644–649, Valencia, Spain. Association for Computational Linguistics.

Claudia Peersman, Walter Daelemans, and Leona Van Vaerenbergh. 2011. Predicting age and gender in online social networks. In *Proceedings of the 3rd international workshop on Search and mining user-generated contents*, pages 37–44.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14, Florence, Italy. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2020. Inexpensive domain adaptation of pretrained language models: Case studies on biomedical NER and covid-19 QA. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1482–1490, Online. Association for Computational Linguistics.

Mohiuddin Md Abdul Qudar and Vijay Mago. 2020. Tweetbert: A pretrained language representation model for twitter text analysis. *arXiv preprint arXiv:2010.11091*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Afshin Rahimi, Timothy Baldwin, and Trevor Cohn. 2017. Continuous representation of location for geolocation and lexical dialectology using mixture density networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 167–176, Copenhagen, Denmark. Association for Computational Linguistics.

Afshin Rahimi, Yuan Li, and Trevor Cohn. 2019. Massively multilingual transfer for NER. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 151–164, Florence, Italy. Association for Computational Linguistics.

Julia Rodina, Yuliya Trofimova, Andrey Kutuzov, and Ekaterina Artemova. 2020. Elmo and bert in semantic change detection for russian. *arXiv preprint arXiv:2010.03481*.

Subendhu Rongali, Abhyuday Jagannatha, Bhanu Pratap Singh Rawat, and Hong Yu. 2020. Improved pretraining for domain-specific contextual embedding models. *arXiv preprint arXiv:2004.02288*.

Maja Rudolph and David Blei. 2017. Dynamic bernoulli embeddings for language evolution. *arXiv preprint arXiv:1703.08052*.

Timo Schick and Hinrich Schütze. 2020a. Exploiting cloze questions for few-shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676*.

Timo Schick and Hinrich Schütze. 2020b. It's not just size that matters: Small language models are also few-shot learners. *arXiv preprint arXiv:2009.07118*.

Dominik Schlechtweg, Anna Hätty, Marco Del Tredici, and Sabine Schulte im Walde. 2019. A wind of change: Detecting and evaluating lexical semantic change across times and domains. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 732–746, Florence, Italy. Association for Computational Linguistics.

Dominik Schlechtweg, Barbara McGillivray, Simon Hengchen, Haim Dubossarsky, and Nina Tahmasebi. 2020. SemEval-2020 task 1: Unsupervised lexical semantic change detection. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1–23, Barcelona (online). International Committee for Computational Linguistics.

Aditya Siddhant, Melvin Johnson, Henry Tsai, Naveen Ari, Jason Riesa, Ankur Bapna, Orhan Firat, and Karthik Raman. 2020. Evaluating the cross-lingual effectiveness of massively multilingual neural machine translation. In *AAAI*, pages 8854–8861.

Wessel Stoop and Antal van den Bosch. 2014. Using idiolects and sociolects to improve word prediction. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 318–327, Gothenburg, Sweden. Association for Computational Linguistics.

Sali A Tagliamonte. 2006. *Analysing sociolinguistic variation*. Cambridge University Press.

Samson Tan, Shafiq Joty, Min-Yen Kan, and Richard Socher. 2020. It's morphin' time! Combating linguistic discrimination with inflectional perturbations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2920–2935, Online. Association for Computational Linguistics.

Xu Tan, Jiale Chen, Di He, Yingce Xia, Tao Qin, and Tie-Yan Liu. 2019. Multilingual neural machine translation with language clustering. 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 963–973, Hong Kong, China. Association for Computational Linguistics.

Abrhalei Tela, Abraham Woubie, and Ville Hautamaki. 2020. Transferring monolingual model to low-resource language: The case of tigrinya. *arXiv preprint arXiv:2006.07698*.

Brian Thompson, Jeremy Gwinnup, Huda Khayrallah, Kevin Duh, and Philipp Koehn. 2019. Overcoming catastrophic forgetting during domain adaptation of 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 2062–2068, Minneapolis, Minnesota. Association for Computational Linguistics.

Ke Tran. 2020. From english to foreign languages: Transferring pre-trained language models. *arXiv preprint arXiv:2002.07306*.

Svitlana Volkova, Theresa Wilson, and David Yarowsky. 2013. Exploring demographic language variations to improve multilingual sentiment analysis in social media. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1815–1827, Seattle, Washington, USA. Association for Computational Linguistics.

Thuy-Trang Vu, Dinh Phung, and Gholamreza Haffari. 2020a. Effective unsupervised domain adaptation with adversarially trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6163–6173, Online. Association for Computational Linguistics.

Tu Vu, Tong Wang, Tsendsuren Munkhdalai, Alessandro Sordoni, Adam Trischler, Andrew Mattarella-Micke, Subhransu Maji, and Mohit Iyyer. 2020b. Exploring and predicting transferability across NLP tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7882–7926, Online. Association for Computational Linguistics.

Zihan Wang, Karthikeyan K, Stephen Mayhew, and Dan Roth. 2020. Extending multilingual BERT to low-resource languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2649–2656, Online. Association for Computational Linguistics.

Walt Wolfram and Natalie Schilling. 2015. *American English: dialects and variation*. John Wiley & Sons.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. 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 833–844, Hong Kong, China. Association for Computational Linguistics.

Shijie Wu and Mark Dredze. 2020. Are all languages created equal in multilingual BERT? In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 120–130, Online. Association for Computational Linguistics.

Ying Xu, Xu Zhong, Antonio Jose Jimeno Yepes, and Jey Han Lau. 2020. Forget me not: Reducing catastrophic forgetting for domain adaptation in reading comprehension. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mT5: A massively multilingual pre-trained text-to-text transformer.

Towards Explainable Dialogue Systems: Explaining Intent Classification using Saliency Techniques

Ratnesh Kumar Joshi[1], **Arindam Chatterjee**[12] **and Asif Ekbal**[1]
[1]Department of Computer Science and Engineering, IIT Patna, Patna, India
[2]Wipro AI Research Labs, Bangalore, India
(ratnesh_1921cs28, arindam_2021cs02, asif)@iitp.ac.in

Abstract

Deep learning based methods have shown tremendous success in several Natural Language Processing (NLP) tasks. The recent trends in the usage of Deep Learning based models for natural language tasks have definitely produced incredible performance for several application areas. However, one major problem that most of these models face is the lack of transparency, *i.e.*, the actual decision process of the underlying model is not explainable. In this paper, first we solve a very fundamental problem of Natural Language Understanding (NLU), *i.e.*, intent detection using a Bidirectional Long Short Term Memory (BiLSTM). In order to determine the defining features that lead to a specific intent class, we use the Layerwise Relevance Propagation (LRP) algorithm to find the defining feature(s). In the process, we conclude that saliency method of ϵLRP (epsilon Layerwise Relevance Propagation) is a prominent process for highlighting the important features of the input responsible for classification of intent, which also provides significant insights into the inner workings, such as the reasons for misclassification by the black box model.

1 Introduction

Chatbots or conversational agents have been gaining immense popularity in recent years. This is one of the most widely used Artificial Intelligence (AI) applications that has a market value of USD 190.8 millions, and is expected to grow upto USD 1,250.1 million by the year 2025[1]. These chatbots are being used in almost every vertical of our society, such as travel, healthcare, judiciary *etc.* With the rapid adaptation of chatbots as digital assistants, it is important that these chatbots should be very robust, as many of these domains (*e.g.*, health, judiciary *etc.*) are very sensitive, and minor inaccuracies in information can lead to significant damage. The model as a whole can be made robust if all its individual components are also accurate. The very first step of most modular dialogue systems is the Natural Language Understanding (NLU) phase, that comprises of *dialogue act classification*, *intent detection* and *slot filling*. This part of the dialogue system plays an important role of deciphering the syntax and semantics of the user input, to aptly produce the bot's reply. While the intent classification focuses on the semantic meaning of the input, slot filling focuses on extracting the relevant information like named entities *etc.*

These systems are not perfect, and even the *state-of-the-art* models very often fail to classify the intent correctly. In order to understand what went wrong in these misclassifications, the features of the text that led to the incorrect classification can provide some helpful information. Due to the rise of deep neural network based architectures the transparency of such models is low. This leads to the requirement of eXplainable Artificial Intelligence (XAI) methods that determine the important features of the input text. There are 2 major methods that highlight the feature importance, namely *saliency* based methods and *attention* based methods. Saliency based methods (Section 2.2) are adhoc techniques that explain individual inference done by the model. This is done after the model training process, hence the cost depends on the number of explanations required. The additional cost of these XAI models tend to make the architectural framework more expensive. Since most of the XAI methods explain each prediction individually, the processing cost keeps on increasing during the model deployment. Attention, on the other hand, calculates the feature importance over the entire training data, and seems like a cost effective alternative to saliency in figuring out the

[1]https://www.grandviewresearch.com/industry-analysis/chatbot-market

relevant features of the input text. Whether attention is actually a good alternative for explanation is still a matter to be explored (Grimsley et al., 2020). When compared to the saliency based techniques, due to the attention's focus on gradient descent as the weight updating criteria, attention has shown high correlation to the gradient based saliency techniques (Jain and Wallace, 2019).

In this paper, we investigate into an explainable deep learning based intent classifier. We compute the features responsible for misclassification of the utterances, in order to get a better idea of why actually the trained model incorrectly predicted these test inputs. This leads to a much better understanding of the limitations of Long Short Term Memory (LSTM) based models. One such limitation is when we go over the ATIS dataset, where we find out that the model misclassifies an Intent class(as shown in figure 2) '*meal*' even though it learns to identify the '*meal*' token as the most important feature as cumulative weight of tokens pointing to the '*flight*' intent is higher. Another such instance can be the '*day_name*' intent being misclassified as '*flight*'. The model does not even learn to pay attention to token(s) like '*day*' or '*day of the week*' as the total instances of '*day_name*' in the entire dataset is less than 0.1% of the dataset.

2 Related Work

In this section, we present a very brief literature survey that starts with a intent classification followed by saliency based explainable models.

2.1 Basic Components of any Conversational System

In practice, two forms of chatbot architectures are prevalent. One being the modular architecture that we focus here in this paper. This procedure breaks the conversational process into a pipeline structure where the upcoming module uses the information gathered from the previous step to build a functional agent. The process includes intent classification, slot filling/entity extraction for the language understanding phase. Dialogue Management (DM) uses the intent and entities extracted to formulate the next action. In this phase we can also employ rules to direct the functioning to a specific action. For example- one rule could be that if the input is exactly the same, use the reply in the training data directly. Of course, this depends on the task at hand. Finally, the information of the DM module is actualized into human understandable text using the Natural Language Generation (NLG) phase. This text generation can be either template based or a neural based, trained on the data available. The second prevalent architecture is the end-to-end architecture which trains a single deep learning model which takes the user input and gives reply utterance directly in one go. Since the entire process of conversation is condensed into one single model this kind of architecture generally requires much more data and since we cannot explicitly impart rules on such a model, it can perform poorly on seemingly simple tasks for a similar amount of data.

2.2 Intent Classification

Intent classification is a highly informational step of any modular dialogue system. It is the initial process of the Natural Language Understanding (NLU) pipeline, which focuses on the prediction of the task the user wants the current input to focus on, from the variety of tasks the model has been trained to perform. The Cambridge dictionary defines intention as 'something you want or plan to do'. Similarly in NLP the intent refers to the task/goal the user wants to accomplish by the conversation. For example, in the user utterance 'what meals are available in flight from Milwaukee to Seattle' the goal/intent of the user is to enquire about the food options. The structure of this utterance is similar to that of a flight search query like 'what flights are available from Milwaukee to Seattle', we want the model to be able to aptly distinguish between these intents. A good intent classifier can bypass poorly directed user queries and correctly processes user intents leading to the smooth conversational flow. Bi-directional Long Short Term Memory (BiLSTM) (Huang et al., 2015) models are a decent baseline in NLP tasks including classification, generation, summarization etc. However, due to the innate opaqueness of neural network based models it leaves a lot to be desired in terms of making the users understand the decision making process. This leads to people using adhoc post-processing steps (saliency techniques) to find out the features/tokens in our text most responsible for the classification output of the model. However, since this adds an overhead to the model, it results in increased cost for providing feature importance.

2.3 Saliency Techniques

Saliency is used in psychology and other fields with subtly varying meanings. For NLP tasks, we refer to saliency as the process of finding the most important features/token(s) responsible for the model. For example, in the utterance ”The service was bad.”, the token(s) ’service’ and ’bad’ are responsible for the utterance to be classified with intent ’complaint’. There are multiple classes of these models with the focus, ranging from token combinations, to game theory concepts (Section 2.3.2) and propagation based (Section 2.3.3). We discuss some techniques for saliency and try to highlight the issues pertaining to these models for adhoc explanations.

2.3.1 Occlusion/Perturbation based

The occlusion or perturbation based methods (Zeiler and Fergus, 2014) compute the feature importance by removing parts of the input and recalculating the classification output and measuring the deviation from the original classification. This deviation from the original prediction then serves as a measure of the importance of the feature with respect to the current model classification. Though these methods are easy to execute for the Natural Language Processing (NLP) tasks, these add a high computation overhead in order to find the important features. For example - for just a text of 10 tokens there can be hundreds of such perturbation based subtexts resulting in a tedious prediction phase. The number of such perturbation based combinations of tokens increases exponentially with the size of the input text. Even though you can use meaningful combination techniques(Fong and Vedaldi, 2017) (here the overhead can be reduced by using many techniques like stopword removal, named entity removal, merging adjective with adverbs such as treating ‘very good’ as a single occlusion candidate etc.) the substantial overhead still exists.

2.3.2 Mathematical Model based

GradientxInput(Denil et al., 2014) calculates saliency of the input text as a function of input sequence vs individual input tokens. Integrated Gradient(Sundararajan et al., 2017) is another gradient based method that extends upon GradientxInput techniques and deals with the sensitivity and implementation invariance. Even though both IG and GradientXInput focus on the sensitivity of the features, it is taken as a measure of the saliency of the input features. SHAP (Lundberg and Lee, 2017) uses the concept of shapley values from game theory to calculate the feature importance.

2.3.3 Propagation based

Layerwise Relevance Propagation (LRP)(Bach et al., 2015) uses an additional backward pass that calculates the relevance of the nodes of our network at each layer. It then uses the weights of the nodes to redistribute the relevance of each layer with respect to the prediction. So, when it finally arrives at the input layer it has the relevant information for each input with regard to the prediction. Since the backward pass flows over the entire network, the cost of saliency is directly proportional to the size of the network trained (example, overhead for a model with 10 layers of depth is more than a model with 2 layers).

2.3.4 Sampling based

LIME(Ribeiro et al., 2016) adopts a local approach to the saliency problem. For a specific input at hand it calculates a locality around the input and then uses that local sample space to train an inherently interpretable model. This newly trained model is then used to make an explanation regarding feature importance for the input. However, in some cases like image classification, even these localities might be too much to be represented by a linear model, Anchors(Ribeiro et al., 2018) is a method which counters this issue by instead forming conditions for prediction. This rule/condition fixes the prediction at local level so changes ant global level. Thus highlighting the parts in the input that are enough to classify it. However, since the technique involves sampling from the training data and also training a new model (both for each explanation to any input), such methods are some of the most expensive saliency methods available.

2.4 Global vs Local Methods

Global methods describe the average behavior of a machine learning model. Global methods estimate expected values based on the distribution of the data. For example, the partial dependence plot (Friedman, 2001), a feature effect plot, is the most expected outcome when all other features are turned insignificant i.e. it shows the marginal effect one or two features have on the predicted outcome of a machine learning model. Since global interpretation methods describe average behavior, they are particularly useful when the user wants to understand the general mechanisms in the data or debug a model.

The counterpart to global methods are local methods. Local interpretation methods explain individual predictions. LIME(Ribeiro et al., 2016) and SHAP(Lundberg and Lee, 2017) are attribution methods, so that the prediction of a single instance is described as the sum of feature effects. Other methods, such as counterfactual explanations(Wachter et al., 2017), are example-based.

For this paper, we focus on local explanations,i.e. look at the individual instances and try to figure the reason for misclassification for each group of instances instead of figuring a general trend for all the misclassification.

3 Methodology

We implement ϵLRP (epsilon Layerwise Relevance Propagation) model over a BiLSTM trained model to find the important features for a particular prediction. This is done with the aim of finding the reason behind the misclassification in incorrectly predicted utterances, as that can potentially help us deduce the reason for misclassification and improve our understanding of the model.

3.1 Intent Classification

For the base model, we use a BiLSTM based architecture. Bidirectional LSTM (BiLSTM) is used to model dependencies on the next time step in the input utterance. These are a combination of a recurrent layers that propagate the sequence forward through blocks and a recurrent module that propagates the sequence backwards through a different block. The tail model uses a concatenation operation on the penultimate two hidden states as input for the final layer.

$$\begin{aligned}
i_{0,t} &= sigmoid(W_i x_t + b_i) \\
\acute{c}_{0,t} &= tanh(W_c x_t + b_c) \\
c_{0,t} &= i_{0,t} \times \acute{c}_{0,t} \\
o_{0,t} &= sigmoid(W_o x_t + V_o c_{0,t} + b_o) \\
h_{0,t} &= o_{0,t} \times tanh(c_{0,t})
\end{aligned}$$

For training, the Adam optimizer (Kingma and Ba, 2014) and categorical cross-entropy loss(Zhang and Sabuncu, 2018) were used. This model had a depth of 2 with each layer having 256 hidden nodes and a dropout of 0.5. We went with a batch size of 24 due to memory restrictions.

3.2 Layer-wise Relevance Propagation

LRP works as an adhoc over the final trained model to calculate the explanation based on the domain of its inputs. LRP takes the weightage of the final classification and distribute this value over the previous layer depending on the contribution/influence of the neuron in the previous layer. This backward pass of sorts recursively distributes the classification weight to the input features, quantifying their importance to the task at hand. For example- if a model predicted the intent to 'flight' with a confidence of 0.8, this 0.8 is then distributed to the neurons of the previous model layer depending on their influence as per equation 1. Recursively this weight/relevance of 0.8 reaches to the feature input layer and the relevance is distributed across the input tokens. For better comprehension we normalize the input token relevance for clarity.

$$\sum_k \frac{a_j w_{j_k}}{\sum_0^j a_j w_{j_k}} R_k \tag{1}$$

Here, j and k denote 2 neurons of consecutive layers, w is weight, and a is the activation. Finally, R denotes the relevance of each neuron. $a_j w_{j_k}$ models the extent of influence of the neuron j in making the neuron k relevant. This influence is then used to distribute the relevance of neuron k to the neurons in the previous layer.R_k is the relevance of the k^{th} neuron at current layer. In this paper, we use ϵLRP which is a modification of base LRP (equation 2) that includes ϵ term in the denominator.

$$\sum_k \frac{a_j w_{j_k}}{\epsilon + \sum_0^j a_j w_{j_k}} R_k \tag{2}$$

The role of ϵ is to still accumulate some relevance even when the influence of the activation of neuron k are weak or contradictory i.e. if R_k is very small then each of the relevance it provides to the neuron(s) j is negligible. The ϵ term helps to maintain mathematical cohesion in case the relevance reaches zero. As ϵ becomes larger, only the most salient explanation factors survive the absorption. This typically leads to explanations that are sparser in terms of input features i.e. the weight distributed is more focused on the important features and all the irrelevant features(stopwords etc) get near zero weightage.This makes the weight distribution less noisy as we can easily focus on the relevant input features. So in conclusion we can say ϵLRP results in sparser and less noisy relevances.

3.3 Model

The model used in this paper uses BiLSTM to train the model for the intent classification task and then employs a LRP model for explanation of the inferences done by the model at testing. The backward pass implemented is a single step process that goes over all the layers of the trained model (from final prediction to the input features) and distributes the weight of the prediction over the input features. Finally, the input feature(s) with highest relative weight are considered responsible for the output of the model.

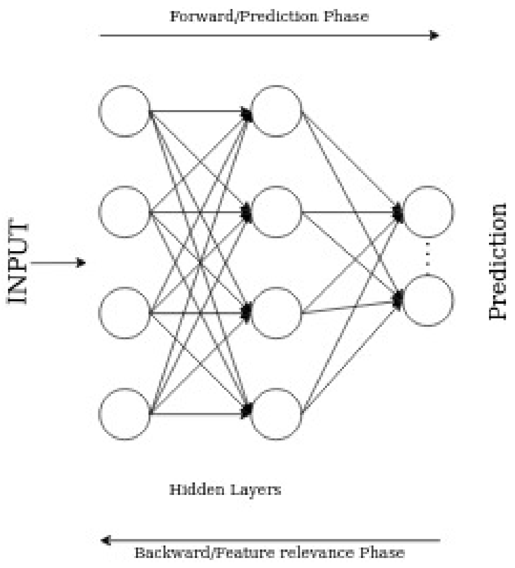


Figure 1: Basic representation of prediction phase (single forward pass) and feature relevance phase (single backward pass) of the BiLSTM + ϵLRP architecture

4 Dataset and Experimental Setup

In this section, we present the details of the datasets and experimental setup.

4.1 Dataset

We use the following benchmark datasets for the experiments.

ATIS: ATIS (Airline Travel Information System) (Hemphill et al., 1990) is a dataset of airline customer service with multiple user utterances and corresponding Intents. The dataset includes 4,478 utterances in training, 500 utterances for validation and 893 utterances in the test set. The data is annotated with 17 intent classes, *viz.* 'flight', 'airfare', 'airline', 'ground_service', 'quantity', 'city', 'abbreviation', 'aircraft', 'distance', 'ground_fare', 'capacity', 'flight_time', 'meal', 'flight_no', 'restriction', 'airport', 'cheapest'. We removed 23 instances labeled with more than one intent.

MultiDoGO: MultiDoGO dataset (Peskov et al., 2019) comprises of six domains, *viz.* airline, fastfood, finance, insurance, media and software. The dataset has two formats, annotated and unannotated. The unannotated version contains 86K conversations, while the annotated version contains 15,000 conversations with 2,500 for each domain. We focus on the user utterances of two sub-domains of airline and finance for intent classification with 38 classes. We use the training, validation and test sets, comprising of 29,742, 4,260 and 8,488 utterances, respectively.

4.2 Experimental Setup

For the experiment, we train the BiLSTM model. We train the model with epochs set to 5, 10 and 20. This is done to avoid any possible overfitting scenario. Adam optimizer and categorical cross-entropy loss were used. To represent the word vectors, a 256 dimensional (non-pretrained) vectors were used. Inference is then generated on the test data, where we primarily focus on the misclassification. The ϵLRP method is then executed on these misclassification to find why these utterances were predicted incorrectly.

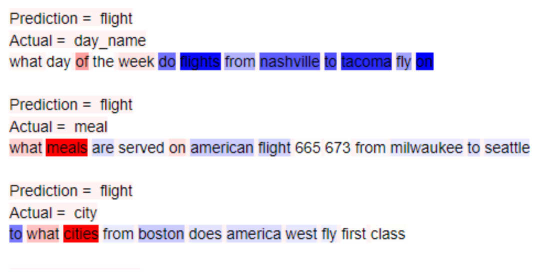
4.3 Results and Analysis

The BiLSTM model is trained on the above mentioned datasets (ATIS airline dataset, MultiDoGO airline subdomain dataset and MultiDoGO finance subdomain dataset). We demonstrate the results in Table 1. Then, ϵLRP is used as an adhoc model to gain insights on misclassification. The interesting cases are highlighted.

Table 1: Results of BiLSTM trained on 3 datasets (ATIS airline dataset, MultiDoGO airline subdomain dataset and MultiDoGO finance subdomain dataset)

Dataset	Accuracy	Precision	Recall
ATIS	0.93	0.93	0.93
MultiDoGO Airline	0.91	0.91	0.91
MultiDoGO Finance	0.89	0.89	0.89

While we closely look at the misclassified cases, we see that most of the misclassifications in the **ATIS dataset** are a result of being predicted as belonging to the 'flight' class instead of the actual class. This can be attributed to the disproportionate training data where 3388 of the 4478 training utterances are for the 'flight' class. This misclassification seems to be due to named entities like locations being taken as a shortcut to classify utterance as intent 'flight'. For example, in Figure 2 one can see that the presence of location tokens (in blue) collectively lead to misclassification as intent 'flight' even though the model knows that the tokens 'meal' and 'cities' (in red) play important role in the classification process (tokens highlighted with blue are responsible for current prediction, the ones with red highlight the second most likely class). This structure of sentences comprise of 4 of the 6 test examples for intent 'meal', all 4 of which are misclassified. The only 2 correctly classified utterances are the ones that do not mention the cities i.e. 'are meals ever served on tower air' and 'are snacks served on tower air'.

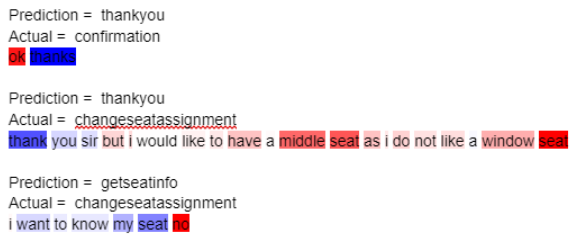

Prediction = flight
Actual = day_name
what day of the week do flights from nashville to tacoma fly on

Prediction = flight
Actual = meal
what meals are served on american flight 665 673 from milwaukee to seattle

Prediction = flight
Actual = city
to what cities from boston does america west fly first class

Figure 2: Examples of misclassifications on the ATIS dataset

For **MultiDoGO airline sub domain**, majority of misclassifications seem to arise from the model paying heavy weightage to the tokens 'ok' as 'confirmation', and thanks for 'thankyou' as the intent class, as shown in Figure 3. There are also a few cases of model being confused between the intents 'getseatinfo' (asking for seat details, ex- I want to know seat no) and 'changeseatassignment' (change the current seating, ex- I want to get window seat) due to having similar tokens in the training data. Also for the utterance 'thank you sir but i would like to have a middle seat as i do not like a window seat' this direct alignment of the word thank to the intent 'thankyou' leads to misclassification even though the model pays attention to the tokens relating to the correct intent 'changeseatassignment'. We found that this association can be lowered by introducing more examples of similar structure to above utterance but that leads to some instances of 'thankyou' intent to 'rejection' intent(ex- thank you so much nothing more bye). For the first example shown the misclassification is negligible in the context that the same utterance 'ok thanks' is labelled as both 'thankyou' and 'confirmation' in the training data.

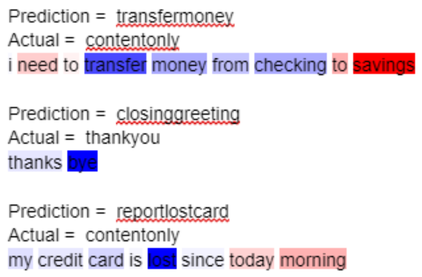

Prediction = thankyou
Actual = confirmation
ok thanks

Prediction = thankyou
Actual = changeseatassignment
thank you sir but i would like to have a middle seat as i do not like a window seat

Prediction = getseatinfo
Actual = changeseatassignment
i want to know my seat no

Figure 3: Examples of misclassifications for the MultiDoGO data

For **MultiDoGO finance dataset**, on the other hand, is filled with misclassifications that seem to be right when going through human evaluation. For example, in Figure 4, examples 1 and 3 can be said to be somewhat correctly predicted (since we are looking at them as individual utterances instead of entire dialogue) even though the actual intents are different. For example 2, the evaluation could go either way depending on preferences of the annotator and the evaluators as all the closing greeting examples have the word thanks in them and are very overlapping in their intention.


Prediction = transfermoney
Actual = contentonly
i need to transfer money from checking to savings

Prediction = closinggreeting
Actual = thankyou
thanks bye

Prediction = reportlostcard
Actual = contentonly
my credit card is lost since today morning

Figure 4: Examples of misclassification on MultiDoGO finance intents

Going through all these datasets, we summarize that there are a lot of inferences that can be drawn with respect to the incorrect classification. The issues arising due to the unbalanced dataset results in forcing the model to pay high attention to some specific tokens. We see the benefits of saliency based methods as it highlights the tokens responsible for the classification. This not only helps us understand the reason for misclassification but also can highlight cases where the data might be incorrectly annotated, resulting in the possibility to improve the quality of dataset along with the classification process.

5 Conclusion and Future work

In this paper, we have attempted to build an explainable intent detection model with the saliency based methods. The model is able to identify the appropriate and relevant features used for intent classification. We also discussed some issues with these approaches, most of which deal with the fact that the saliency techniques calculate the feature importance (which constitute an explanation) as an adhoc measure.

Saliency does have quite a few benefits of itself. The modular nature of the implementations provides a degree of model-agnostic behaviour which allows us to treat the black boxed nature of the deep learning models as an afterthought and focus entirely on the performance. After the model is trained and tuned, we applied the saliency techniques for determining the feature relevance. This also ensures that we can apply different saliency techniques for the same base model and the same saliency technique to different models allowing for a high degree of robustness.

However, even for saliency it is not necessary that the importance assigned to a feature is, in fact, due to the relevance of the features but could simply be a result of the underlying issues with the technique used. For example, in occlusion based methods, if we remove a feature, it is possible that the change in the prediction is just the result of the new input not being in the format the model expects(Kindermans et al., 2019).

For future work, we plan to use the feature importance information and use it to retrain the model in such a way to reduce misclassification. One such method could be to use the important features of misclassifications to help identify which kind of data to add, to improve the performance of the model further. However, such a method needs to be done in such a way that the incorrect predictions do not get corrected at the cost of misclassification of originally correctly predicted utterances. Another such method could be to identify the nodes which are more relevant to a highly misclassified intent and boost those neurons to improve model performance. However, this also needs to make sure the nodes we are associating with a particular intent do not have high influence on other intents as well, as that might lower the accuracy of some other intent.

6 Acknowledgement

Ratnesh Joshi gratefully acknowledge the UGC-JRF NET Fellowship (No. 3601/(NET-JULY2018)). Authors acknowledge the generous support of IMPRINT-2 Project "Sevak-An Intelligent Indian Language Chatbot" Grant No-IMP/2018/002072).

References

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140.

Misha Denil, Alban Demiraj, and Nando De Freitas. 2014. Extraction of salient sentences from labelled documents. *arXiv preprint arXiv:1412.6815.*

Ruth C Fong and Andrea Vedaldi. 2017. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE international conference on computer vision*, pages 3429–3437.

Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Christopher Grimsley, Elijah Mayfield, and Julia R.S. Bursten. 2020. Why attention is not explanation: Surgical intervention and causal reasoning about neural models. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1780–1790, Marseille, France. European Language Resources Association.

Charles T Hemphill, John J Godfrey, and George R Doddington. 1990. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990.*

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991.*

Sarthak Jain and Byron C Wallace. 2019. Attention is not explanation. *arXiv preprint arXiv:1902.10186.*

Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. 2019. The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280. Springer.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, pages 4768–4777.

Denis Peskov, Nancy Clarke, Jason Krone, Brigi Fodor, Yi Zhang, Adel Youssef, and Mona Diab. 2019. Multi-domain goal-oriented dialogues (multidogo): Strategies toward curating and annotating large scale dialogue data. 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 4526–4536.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.

Zhilu Zhang and Mert R Sabuncu. 2018. Generalized cross entropy loss for training deep neural networks with noisy labels. In *32nd Conference on Neural Information Processing Systems (NeurIPS)*.

Comparing in context: Improving cosine similarity measures with a metric tensor

Isa M. Apallius de Vos [*]
{i.m.apalliusdevos}
Ghislaine L. van den Boogerd [*]
{g.l.vandenboogerd}
Mara D. Fennema [*]
{m.d.fennema}
@students.uu.nl
Utrecht University, The Netherlands

Adriana D. Correia [†]
a.duartecorreia@uu.nl
Utrecht University, The Netherlands

Abstract

Cosine similarity is a widely used measure of the relatedness of pre-trained word embeddings, trained on a language modeling goal. Datasets such as WordSim-353 and SimLex-999 rate how similar words are according to human annotators, and as such are often used to evaluate the performance of language models. Thus, any improvement on the word similarity task requires an improved word representation. In this paper, we propose instead the use of an extended cosine similarity measure to improve performance on that task, with gains in interpretability. We explore the hypothesis that this approach is particularly useful if the word-similarity pairs share the same context, for which distinct contextualized similarity measures can be learned. We first use the dataset of Richie et al. (2020) to learn contextualized metrics and compare the results with the baseline values obtained using the standard cosine similarity measure, which consistently shows improvement. We also train a contextualized similarity measure for both SimLex-999 and WordSim-353, comparing the results with the corresponding baselines, and using these datasets as independent test sets for the all-context similarity measure learned on the contextualized dataset, obtaining positive results for a number of tests.

1 Introduction

Cosine similarity has been largely used as a measure of word relatedness, since vector space models for text representation appeared to automatically optimize the task of information retrieval (Salton and McGill, 1983). While other distance measures are also commonly used, such as Euclidean distance (Witten et al., 2005), for cosine similarity only the vector directions are relevant, and not their norms. More recently, pre-trained word representations, also referred to as embeddings, obtained from neural network language models, starting from word2vec (W2V) (Mikolov et al., 2013), emerged as the main source of word embeddings, and are subsequently used in model performance evaluation on tasks such as word similarity (Toshevska et al., 2020). Datasets such as SimLex-999 (Hill et al., 2015) and WordSim-353 (Finkelstein et al., 2001), which score similarity between word-pairs according to the assessment of several humans annotators, have become the benchmarks for the performance of a certain type of embedding on the task of word similarity (Recski et al., 2016; Dobó and Csirik, 2020; Speer et al., 2017; Banjade et al., 2015).

For $\vec{n}_a$ and $\vec{n}_b$, the vector representations of two distinct words w_a and w_b, cosine similarity takes the form

$$cos_{ab} = \frac{\vec{n}_a \cdot \vec{n}_b}{||\vec{n}_a||\ ||\vec{n}_b||}, \tag{1}$$

with the Euclidean *inner product* between any two vectors $\vec{n}_a$ and $\vec{n}_b$ given as

$$\vec{n}_a \cdot \vec{n}_b = \sum_i \vec{n}_a^i \vec{n}_b^i, \tag{2}$$

and the *norm* of a vector $\vec{n}_a$ given as

$$||\vec{n}_a|| = \sqrt{\vec{n}_a \cdot \vec{n}_a}, \tag{3}$$

dependent on the inner product (Axler, 1997).

Using this measure of similarity, improvements can only take place if the vectors that represent the words change. However, the assumption that the vectors interact using a Euclidean inner product becomes less plausible when it comes to higher order vectors. If, differently, we consider that the vector components are not described in a Euclidean basis, then we enlarge the possible relationships

[*]These authors contributed equally to this work.
[†]Corresponding author.

between the vectors. Specifically in the calculation of the inner product, on which the cosine similarity depends, we can use an intermediary *metric* tensor. By challenging the assumption that the underlying metric is Euclidean, cosine similarity values can be improved *without changing vector representations*.

We identify two main motivations to search for improved cosine similarity measures. The first motivation has to do with the cost of training larger and more refined language models (Bender et al., 2021). By increasing the performance on a task simply by changing the evaluation measure without changing the pre-trained embeddings, we expect that better results can be achieved with more efficient and interpretable methods. This is particularly true of contextualized datasets, with benefits not only for tasks such as word similarity, but also others that use cosine similarity as a measure of relatedness, such as content based recommendation systems (Schwarz et al., 2017), and where it can be particularly interesting to explore the different metrics that emerge as representations of vector relatedness.

The second motivation comes from compositional distributional semantics, where words of different syntactic types are represented by tensors of different ranks, and representations of larger fragments of text are produced via tensor contraction (Coecke and Clark, 2010; Grefenstette and Sadrzadeh, 2011a,b; Milajevs et al., 2014; Baroni et al., 2014; Paperno et al., 2014). This framework has proved to be a valuable tool for low resource languages, enhancing the scarce available data with a grammatical structure for composition, providing embeddings of complex expressions (Abbaszadeh et al., 2021). As these contractions depend on an underlying metric that is usually taken to be Euclidean, improvements have only been achieved, once again, by modifying word representations (Wijnholds and Sadrzadeh, 2019). As proposed by Correia et al. (2020), another way to improve on these results consists in using a different metric to mediate tensor contractions. Metrics obtained in tasks such as word similarity can be transferred to tensor contraction, and thus we expect this work to open new research avenues on the compositional distributional framework, providing a better integration with (contextual) language models.

This paper is organized as follows. In §2 we introduce an extended cosine similarity measure, motivating the introduction of a metric on the hypothesis that it can optimize the relationships between the vectors. In §3 we explain our experiment on contextualized and non-contextualized datasets to test whether improvements can be achieved. In §4 we present the results obtained in our experiments and in §5 we discuss these results and propose further work.

Our contributions are summarized below:

- Use of contextualized datasets to explore contextualized dynamic embeddings and evaluate the viability of contextualized similarity measures;

- Expansion of the notion of cosine similarity, motivating our model theoretically, contributing to a conceptual simplification that yields interpretable improvements.

1.1 Related Literature

Variations on similarity metrics on the contextualized dataset of Richie et al. (2020) have been first explored in Richie and Bhatia (2021), but only on static vector representations and diagonal metrics. Other analytical approaches to similarity learning have been identified in Kulis et al. (2013). The notion of soft cosine similarity of Sidorov et al. (2014) presents a relevant extension theoretically similar to ours, but motivated and implemented differently. Using count-base vector space models with words and n-grams as features, the authors extract a similarity score between features, using external semantic information, that they use as a distance matrix that can be seen as a metric; however, they do not implement it as in Eq. (4), but instead they transform the components by creating a higher dimensional vector space where each entry is the average of the components in two features, multiplied by the metric, whereas we, by contrast, learn the metric automatically and apply it to the vectors directly. Hewitt and Manning (2019) also use a modified metric for inner product to probe the syntactic structure of the representations, showing that syntax trees are embedded implicitly in deep models' vector geometry.

Context dependency in how humans evaluate similarity, which we based our study on, has been widely supported in the psycholinguistic literature. Tversky (1977) shows that similarity can be expressed as a linear combination of properties of objects, Barsalou (1982) looks at how context-dependent and context-independent properties influence similarity perception, Medin et al. (1993) explore how similarity judgments are constrained by the very fact of being requested, and Goldstone et al. (1997) test how similarity judgments are influenced by context that can either be explicit or perceived.

2 Model

A metric is a tensor that maps any two vectors to an element of the underlying field $\mathbb{K}$, which in this case will be the field of real numbers $\mathbb{R}$. This element is what is known as the *inner product*. To this effect, the metric tensor can be represented as a function, not necessarily linear, over each of the coordinates of the vectors it acts on. In geometric terms, the metric characterizes the underlying geometry of a vector space, by describing the projection of the underlying manifold of a non-Euclidean geometry to a Euclidean geometry $\mathbb{R}^n$ (Wald, 2010). The inner product between two vectors is informed by the metric in a precise way, and is representative of how the distance between two vectors should be calculated.

A standard example consists of two unit vectors on a sphere, which is an $\mathbb{S}^2$ manifold that can be mapped onto $\mathbb{R}^3$. If the vectors are represented in spherical coordinates, which are a map from $\mathbb{S}^2$ to $\mathbb{R}^3$, the standard method of computing the angle between the vectors using Eq. (1) will fail to give the correct value. The vectors need to be transformed by the appropriate non-linear metric to the Euclidean basis in $\mathbb{R}^3$ before a contraction of the coordinates can take place. To illustrate this, take as an example a triangle drawn on the surface of a sphere $\mathbb{S}^2$. If it is projected onto a planisphere $\mathbb{R}^3$, a naive measurement of its internal angles will exceed the known 180 degrees, which corresponds to a change in the inner product between the vectors tangents to the triangle corners (see Levart (2011) for a demonstration). To preserve this inner product, and thus recover the equivalence between a triangle on a spherical surface and a triangle on a Euclidean plane, the coordinates need to be properly transformed by the appropriate metric before they are contracted.

By the same token, we explore here the possibility that the shortcomings of the values obtained using cosine similarity when compared with human similarity ratings are not due to poor vector representations, but to a measure that fails to assess the distance between the vectors adequately. To test this hypothesis, we generalize the inner product of Eq. (2) to accommodate a larger class of relationships between vectors, modifying it using a metric represented by the distance matrix d, once a basis is assumed, that defines the inner product between two vectors as

$$\vec{n}_a \cdot_d \vec{n}_b = \sum_{ij} \vec{n}_a^i d^{ij} \vec{n}_b^j, \tag{4}$$

where $\vec{n}_a^i$ is the ith component of $\vec{n}_a$. Using a metric of this form, the best we can achieve is a linear rescaling of the components of the vectors, which entails the existence of a non-orthogonal basis. The metric d is required to be bilinear and symmetric, which is satisfied if

$$d^{sym} = B^T B, \tag{5}$$

such that Eq. (4) can be rewritten as

$$\vec{n}_a \cdot_d \vec{n}_b = (B\vec{n}_a)^T \cdot (B\vec{n}_b)\,. \tag{6}$$

We can thus learn the components of a metric for a certain set of vectors by fitting it to the goal of preserving a specified inner product. In the case of word similarity, the matrix B can be learned supervised on human similarity judgments, towards the goal that a contextualized cosine similarity applied to a set of word embeddings, using Eq. (6), returns the correct human assessment. An advantage of this approach is that the cosine is symmetric with respect to its inputs, which is a nice property that this extension preserves by requiring that symmetry of the metric.

3 Methods

The general outline of our experiment is as follows. First, we learn contextualized cosine similarity measures for related (contextualized) pairs of words, and afterwards for unrelated (non-contextualized) pairs of words. A schematic representation can be found in Fig. 1. We then test whether these learned measures are transferable and provide improvements on word pairs that were not seen during training, when compared with the standard cosine similarity baseline.

3.1 Datasets

For a contextualized assessments of word similarity, we use the dataset of Richie et al. (2020), where 365 participants were asked to judge the similarity between English word-pairs that are co-hyponyms of eight different hypernyms (Table 1). Participants were assigned a specific hypernym and were asked

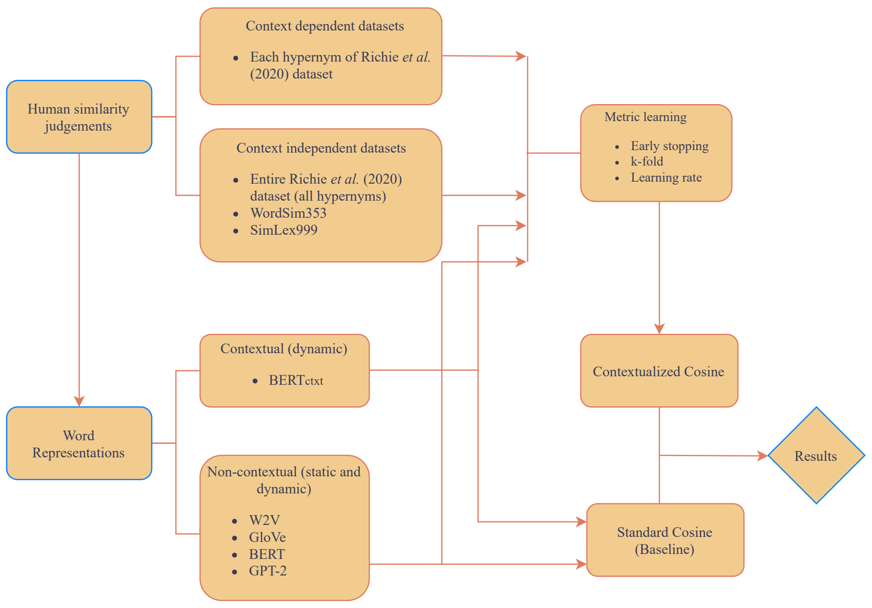

Context dependent datasets
Each hypernym of Richie et al. (2020) dataset
Human similarity judgements
Context independent datasets
Entire Richie et al. (2020) dataset (all hypernyms)
WordSim353
SimLex999
Metric learning
Early stopping
k-fold
Learning rate
Contextual (dynamic)
BERTctxt
Contextualized Cosine
Word Representations
Results
Non-contextual (static and dynamic)
W2V
GloVe
BERT
GPT-2
Standard Cosine (Baseline)

Figure 1: Schematic representation of the experiment leading up to the results in Tables 4 and 5.

to rate the similarity between each co-hyponym pair from 1 to 7, with the highest rating indicating the words to be maximally similar. The number of annotators varies per hypernym, but each word-pair is rated by around 30 annotators, such that for the largest categories each annotator only saw a fraction of the totality of the word-pairs. As examples from the hypernym ‘Clothing’, the word-pair ‘hat/overalls’ was rated by 32 of the 61 annotators, resulting in an average similarity of 1.469, while ‘coat/gloves’ had an average similarity rating of 3.281 and ‘coat/jacket’ of 6.438, also by 32 annotators. The average similarity was computed for all word-pairs and rescaled to a value between 0 and 1, to be used as the target for supervised learning.

Besides trying to fit a contextualized similarity measure to each hypernym, we also considered the entire all-hypernyms dataset, in order to test whether training on the hypernyms separately would result in a better cosine measure compared with when the hypernym information was disregarded.

To test whether similarity measures can be learned if the similarity of words is not assessed within a specific context, we use the WordSim-353 (WS353) (Finkelstein et al., 2001) and part of the SimLex-999 (SL999) (Hill et al., 2015) datasets, where the word-pairs bear no specific semantic relation. From the SL999 dataset only the nouns were included, resulting in a dataset of 666 word-pairs. Additionally, we use these datasets to verify whether the similarity metric learned by training on the whole dataset of Richie et al. (2020) can be transferred to other, more general, datasets.

Table 1: Number of words, word-pairs and human annotators per hypernym.

Hypernym	Words	Pairs	Annotators
Birds	30	435	54
Clothing	29	406	61
Professions	28	378	67
Sports	28	378	61
Vehicles	22	231	28
Fruit	21	210	31
Furniture	20	190	33
Vegetables	20	190	30
All	198	2418	365

3.2 Word embeddings

To fine-tune the cosine similarity measure, we start from different pre-trained word representations. We do that for two classes of embeddings, static and dynamic.

Static embeddings were obtained from a pre-trained word2vec (W2V) model (Mikolov et al., 2013) and a pre-trained GloVe model (Pennington et al., 2014), each used to encode each word in the pair. Dynamic embeddings were obtained from two Transformers-based models, pre-trained BERT (Devlin et al., 2019) and GPT-2 models (Radford et al.,

Representation	Corpus	Corpus size	Dim
word2vec	Google News	100B	300
GloVe	GigaWord Corpus & Wikipedia	6B	200
BERT$_{\text{base-uncased}}$	BooksCorpus & English Wikipedia	3.3B	768
GPT-2$_{\text{medium}}$	8 million web pages	$\sim$ 40 GB	768

Table 2: Pre-trained embeddings obtained from different source language models, with BERT and GPT-2 implemented using the Huggingface Transformers library.

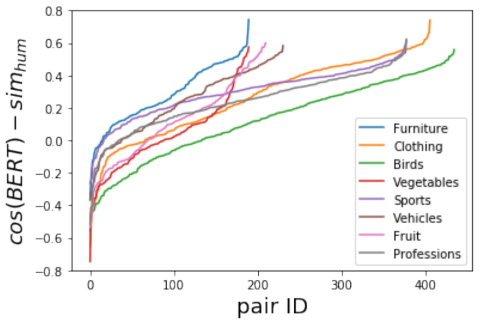
Hypernym	Context words
Birds	`small, migratory, other, water, breeding`
Clothing	`cotton, heavy, outer, winter, leather`
Professions	`health, legal, engineering, other, professional`
Sports	`youth, women, men, ea, boys`
Vehicles	`military, agricultural, motor, recreational, commercial`
Fruit	`citrus, summer, wild, sweet, passion`
Furniture	`wood, furniture, modern, antique, office`
Vegetables	`some, wild, root, fresh, green`

Table 3: Five most likely words for masked token preceding hypernym token using BERT.

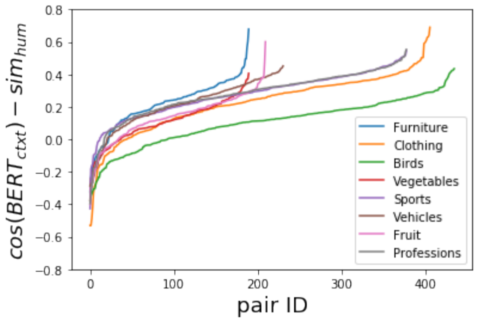
2019) (see Table 2). Here the representation of each word was taken to be the average representation of sub-word tokens when necessary, excluding the [CLS] and [SEP] tokens.

The token representations provided by the BERT model, as a bidirectional dynamic language model, can change depending on the surrounding context tokens. As such, additional contextualized embeddings were retrieved, BERT$_{ctxt}$, to test whether performance could be improved relative to the baseline cosine metric by using the hypernym information, as well as when compared with the hypernym cosine metric learned on non-contextualized representations. In this way we test whether leveraging the contextual information intrinsic to this dataset can in itself improve similarity at the baseline level, without the need of further training.

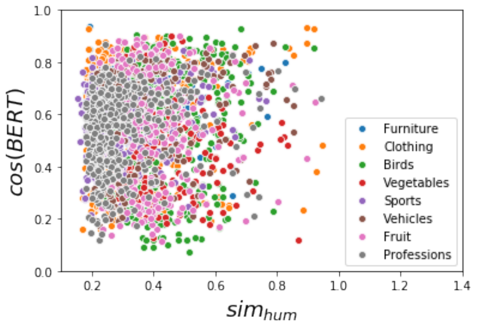
The contextualized vectors of BERT$_{ctxt}$ were obtained by first having BERT predict the five most likely adjectives that precede each hypernym using (`[MASK] <hypernym>`), and then using those adjectives to obtain five contextualized embeddings for each co-hyponym, subsequently averaged over. Most of the predicted words were adjectives, and the few cases that were not were filtered out. For instance, for the category 'Clothing', the most likely masked tokens were 'cotton', 'heavy', 'outer', 'winter' and 'leather'. The contextualized representation of each hyponyms of 'Clothing' was



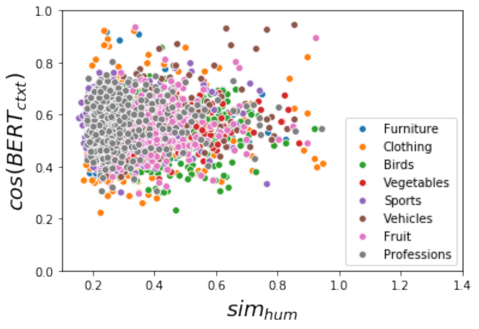
(a)



(b)



(c)



(d)

Figure 2: Distributions of pairwise human similarity judgments sim_{hum} and cosine similarity measures using either BERT representations ($\cos$(BERT)) or contextualized BERT representations ($\cos$(BERT$_{ctxt}$)). In (a) and (b) the absolute difference of scores, ordered per hypernym, is shown, while (c) and (d) represent the distribution of different similarity scores with respect to each other. Comparing the first two plots we can see a regularization effect by contextualizing the representations, and between the last two plots we can see a clustering effect.

thus calculated as its average representation in the context of each of the adjectives, so that, for instance, for ’coat’ we first obtained its contextualized representation in ‘cotton coat’, ‘heavy coat’, ‘outer coat’, ‘winter coat’, and ‘leather coat’, performing a final averaging. The full list of context words can be found in Table 3. Figs. 2a and 2b show that this transformation reduces the absolute extreme values of the difference between the values of the standard cosine similarity and the corresponding human similarity assessments, while regularizing the bulk of the differences closer to the desired value of 0. We tested other forms of contextualizing, such as (`<hypernym> is/are [MASK]`), but the resulting representations did not show as much improvement.

The WS353 and SL999 datasets were only trained with non-contextualized embeddings, since we cannot obtain contextualized embeddings for the nouns in these datasets using the same method. For consistency, the models that were learned with contextualized representations were not tested on these datasets at the final step of our experiment.

3.3 Model

A linear model was implemented on the PyTorch machine learning framework to learn the parameters of B, without a bias, such that a word initially represented by input$_a$ is transformed to input’$_a$ = Binput$_a$. The forward function of this model takes two inputs and returns

$$\frac{(\text{input'}_a)^T \cdot \text{input'}_b}{\sqrt{(\text{input'}_a)^T \cdot \text{input'}_1}\sqrt{(\text{input'}_b)^T \cdot \text{input}_b}}, \quad (7)$$

where a and b correspond to the indices of the words of a given word-pair[1].

3.4 Cross-validation

The number of co-hyponyms per hypernym is small when compared with the number of parameters in B to be trained, which depends on the square of the dimension **Dim** of each representation. To ensure that the models did not overfit, a k-fold cross-validation was used during training (Raschka, 2015), which divided each dataset in k training sets and non-overlapping development sets. Additionally, early stopping of training was implemented in the event that the validation loss increased for ten consecutive epochs after it dropped below 0.1 (Bishop, 2006).

[1]`https://github.com/maradf/Contextualized-Cosine`

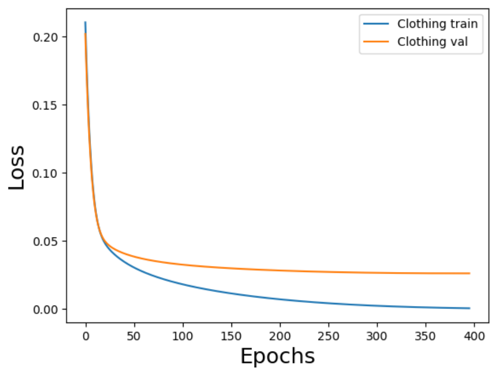


Figure 3: Example of learning curve, showing losses over epochs, from a fold training on the hypernym **Clothes** on the GloVe embeddings. In this case, training was stopped early at 397 epochs.

3.5 Hyperparameter selection

Per each dataset h (each hypernym, all hypernyms, WS353 or SL999) and learning rate l_r, k models B^h_{i,l_r} were trained, with $i \in \{1, ..., k\}$ and with k corresponding validation sets val_i. The training was done using two 16 cores (64 threads) Intel Xeon CPU at 2.1 GHz.

A fixed seed was used to find the best combination of the learning rate l_r (1×10^{-5}, 1×10^{-6}, and 1×10^{-7}) and the number of folds (5, 6 and 7) for the k-fold cross-validation. The regression to the best metric was done using the mean square error loss function and the Adam optimizer. The maximum number of training epochs was set to 500, as most models converged at that point as per preliminary learning curve inspection (Fig.3). The implementation of early stopping resulted in *de facto* variation of the number of epochs required to train each model.

3.6 Testing the model

Each one of the B^h_{i,l_r} models was tested on the corresponding holdout validation set val_i, resulting in two correlation scores between the models’ predicted similarity scores and the human judgment scores: a Pearson correlation score $r^h_{i,l_r}(val^h_i)$ and a Spearman correlation score $\rho^h_{i,l_r}(val^h_i)$. A final score per k and l_r was calculated using the average performance on the validation sets as

(a) Pearson correlations.

Dataset (h)	BERT		BERT$_{ctxt}$		GPT-2		word2vec		GloVe	
	Model	Base	Model	Base	Model	Base	Model	Base	Model	Base
Birds	**0.311**	0.098	**0.316**	0.042	**0.200**	-0.023	**0.293**	0.213	**0.215**	0.194
Clothing	**0.550**	0.141	**0.515**	0.065	**0.501**	0.349	**0.529**	0.417	**0.574**	0.364
Professions	**0.501**	0.193	**0.601**	0.073	**0.651**	0.542	**0.635**	0.566	0.529	0.529
Sports	**0.452**	0.175	**0.543**	0.139	**0.556**	0.324	**0.532**	0.418	**0.580**	0.386
Vehicles	**0.496**	0.218	**0.616**	0.123	**0.645**	0.385	**0.738**	0.719	**0.703**	0.567
Fruit	**0.315**	0.016	**0.378**	-0.037	**0.333**	0.203	**0.361**	0.239	**0.571**	0.392
Furniture	**0.353**	-0.018	**0.539**	-0.035	**0.568**	0.399	**0.368**	0.333	**0.470**	0.462
Vegetables	**0.211**	-0.059	**0.293**	-0.044	**0.378**	0.144	**0.577**	0.281	**0.562**	0.290
All hypernyms	**0.434**	0.100	**0.542**	0.040	**0.508**	0.287	**0.483**	0.400	**0.539**	0.397
WordSim-353	**0.517**	0.238	-	-	**0.651**	0.647	0.637	0.654	**0.622**	0.568
SimLex-999	**0.403**	0.161	-	-	**0.555**	0.504	**0.495**	0.455	**0.510**	0.408

(b) Spearman correlations.

Dataset (h)	BERT		BERT$_{ctxt}$		GPT-2		word2vec		GloVe	
	Model	Base	Model	Base	Model	Base	Model	Base	Model	Base
Birds	**0.260**	0.102	**0.299**	0.052	**0.190**	-0.054	**0.250**	0.211	**0.238**	0.201
Clothing	**0.436**	0.184	**0.467**	0.059	**0.445**	0.276	**0.510**	0.414	**0.513**	0.384
Professions	**0.501**	0.248	**0.578**	0.170	**0.560**	0.473	**0.518**	0.410	0.482	0.486
Sports	**0.391**	0.174	**0.526**	0.142	**0.540**	0.291	**0.458**	0.339	**0.478**	0.325
Vehicles	**0.518**	0.238	**0.601**	0.056	**0.626**	0.288	**0.709**	0.687	**0.680**	0.596
Fruit	**0.265**	-0.014	**0.333**	-0.103	**0.365**	0.173	**0.368**	0.277	**0.491**	0.342
Furniture	**0.353**	-0.032	**0.491**	-0.120	**0.527**	0.393	**0.442**	0.402	**0.464**	0.451
Vegetables	**0.217**	-0.028	**0.305**	0.015	**0.363**	0.089	**0.587**	0.290	**0.528**	0.228
All hypernyms	**0.407**	0.111	**0.504**	0.034	**0.504**	0.242	**0.446**	0.379	**0.477**	0.377
WordSim-353	**0.543**	0.267	-	-	**0.715**	0.705	0.675	0.701	**0.624**	0.579
SimLex-999	**0.416**	0.180	-	-	**0.566**	0.513	**0.475**	0.445	**0.500**	0.374

Table 4: Best correlation scores between human similarity judgments and similarity scores found by the trained model, compared with baseline cosine metric values of the same hyperparameters. The underlined correlation values are the statistical significant values with a $p < 0.05$, and the bold values correspond to model correlations that were higher than base correlations.

$$r^h_{k,l_r} = \frac{1}{k}\sum_{i=1}^{k} r^h_{i,l_r}(val^h_i), \tag{8}$$

$$\rho^h_{k,l_r} = \frac{1}{k}\sum_{i=1}^{k} \rho^h_{i,l_r}(val^h_i). \tag{9}$$

The baseline results were obtained in a similar form, but with the model B^{std} corresponding to the identity matrix, returning the standard cosine similarity rating as

$$r^{h,std}_k = \frac{1}{k}\sum_{i=1}^{k} r^{std}(val^h_i), \tag{10}$$

$$\rho^{h,std}_k = \frac{1}{k}\sum_{i=1}^{k} \rho^{std}(val^h_i). \tag{11}$$

The model results shown in Table 4 correspond to the best correlation values obtained using Eqs. (8) and (9), with the baselines given as in Eqs. (10) and (11). The hyperparameters corresponding to the best results can be found in Table 5, along with the relative change in correlation performance. As the seed was fixed, the differences in performance achieved by models trained on each hypernym and on all-hypernyms of the contextualized dataset were not due to randomization errors. The final correlation per fold on the entire all-hypernyms dataset was found by first calculating the correlation per hypernym and then averaging over all eight hypernyms.

To test the transferability of the metric learned on the all-hypernyms dataset to other datasets, the model that returned the best correlation scores on the validation datasets of the all-hypernyms dataset was tested on the entire WS353 and SL999 datasets. As the best performing model consists in fact of k models, each one of these was tested on the entire datasets, as

(a) Pearson correlations.

Dataset (h)	BERT		BERT$_{ctxt}$		GPT-2		W2V		GloVe	
	%	l_r, k	%	lr, k	%	lr, k	%	lr, k	%	lr, k
Birds	217	10^{-6},5	652	10^{-6}, 5	**770**	10^{-5}, 5	38	10^{-5}, 5	11	10^{-5}, 7
Clothing	290	10^{-6},5	**692**	10^{-6}, 6	44	10^{-5}, 6	27	10^{-5}, 7	58	10^{-6}, 5
Professions	160	10^{-6}, 5	**723**	10^{-6}, 6	20	10^{-5}, 5	12	10^{-5}, 7	0	10^{-5}, 5
Sports	158	10^{-5}, 6	**291**	10^{-6}, 6	72	10^{-5}, 6	27	10^{-5}, 6	50	10^{-6}, 7
Vehicles	128	10^{-6}, 6	**401**	10^{-5}, 7	68	10^{-5}, 5	3	10^{-5}, 5	24	10^{-6}, 6
Fruit	**1869**	10^{-5}, 7	922	10^{-6}, 6	64	10^{-5}, 7	51	10^{-6}, 5	46	10^{-7}, 7
Furniture	**1861**	10^{-5}, 7	1440	10^{-6}, 6	42	10^{-5}, 7	11	10^{-5}, 6	2	10^{-5}, 6
Vegetables	258	10^{-5}, 7	**566**	10^{-6}, 6	163	10^{-5}, 5	105	10^{-6}, 7	94	10^{-6}, 5
All	334	10^{-5}, 5	**1255**	10^{-6}, 7	77	10^{-5}, 6	21	10^{-5}, 6	36	10^{-7}, 6
WordSim-353	**117**	10^{-6}, 7	-	-	1	10^{-5}, 7	-3	10^{-5}, 6	10	10^{-5}, 5
SimLex-999	**150**	10^{-6}, 7	-	-	10	10^{-5}, 6	9	10^{-5}, 6	25	10^{-6}, 5

(b) Spearman correlations.

Dataset (h)	BERT		BERT$_{ctxt}$		GPT-2		W2V		GloVe	
	%	lr, k	%	lr, k	%	lr, k	%	lr, k	%	lr, k
Birds	155	10^{-6}, 5	**475**	10^{-6}, 5	252	10^{-5}, 7	18	10^{-5}, 5	18	10^{-7}, 5
Clothing	137	10^{-6}, 5	**692**	10^{-6}, 6	61	10^{-5}, 7	23	10^{-5}, 7	34	10^{-6}, 5
Professions	102	10^{-6}, 7	**240**	10^{-6}, 5	18	10^{-5}, 5	26	10^{-5}, 7	-1	10^{-7}, 6
Sports	125	10^{-5}, 6	**270**	10^{-6}, 6	86	10^{-5}, 6	35	10^{-5}, 6	47	10^{-6}, 6
Vehicles	118	10^{-6}, 6	**973**	10^{-6}, 6	117	10^{-5}, 7	3	10^{-5}, 5	14	10^{-6}, 6
Fruit	**1793**	10^{-6}, 7	223	10^{-6}, 6	111	10^{-5}, 6	33	10^{-6}, 6	44	10^{-7}, 7
Furniture	**1003**	10^{-6}, 6	309	10^{-6}, 5	34	10^{-5}, 5	10	10^{-5}, 6	3	10^{-6}, 7
Vegetables	675	10^{-5}, 7	**1933**	10^{-6}, 6	308	10^{-5}, 5	102	10^{-6}, 7	132	10^{-6}, 5
All hypernyms	267	10^{-5}, 5	**1382**	10^{-6}, 7	108	10^{-5}, 6	18	10^{-5}, 6	27	10^{-6}, 5
WordSim-353	**103**	10^{-6}, 5	-	-	1	10^{-5}, 7	-4	10^{-6}, 5	8	10^{-5}, 5
SimLex-999	**131**	10^{-6}, 7	-	-	10	10^{-5}, 6	7	10^{-5}, 6	34	10^{-6}, 5

Table 5: Change (%) in correlation from Table 4, given by $(|\text{Model}| - |\text{Base}|)/|\text{Base}|$, at corresponding best hyperparameters (lr, k). Values in bold indicate the highest increase on a given dataset.

$$r^{h,test}_{k,l_r} = \frac{1}{k}\sum_{i=1}^{k} r^{All-hyp}_{i,l_r}(test^h), \quad (12)$$

$$\rho^{h,test}_{k,l_r} = \frac{1}{k}\sum_{i=1}^{k} \rho^{All-hyp}_{i,l_r}(test^h), \quad (13)$$

with $h \in \{\text{WS353}, \text{SL999}\}$.

The baselines for these results were obtained by applying B^{std} to the entire WS353 and SL999 datasets as

$$r^{h,std} = r^{std}(test^h), \quad (14)$$

$$\rho^{h,std} = \rho^{std}(test^h). \quad (15)$$

As the correlation functions are not linear, the results from Eqs. (10) and (11) for the WS353 and SL999 datasets are expected to differ from those obtained using Eqs. (14) and (15) for the same datasets.

4 Results

The validation results on Table 4 show consistent improvements over the baselines, with statistical significance. This confirms that the modification introduced to the cosine measure worked in a principled way, and consistent with the results found by Richie and Bhatia (2021). On the individual hypernym datasets, 'Vehicles' showed the best correlations, except for the Pearson correlation in GPT-2, in spite of not being the largest hypernym dataset. On the contrary, the smallest categories showed the lowest correlations. In general, the relative performance of hypernyms according to the baselines extends to the model correlations, although with better performance. With some exceptions, mainly in the 'Birds' hypernym, the best performing representation was GPT-2, followed by W2V, but the relative increase as shown in Table 5 was clearly superior for the dynamic representations. An important observation that we make is that the model trained on all hypernyms had a better performance than the average performance on the individual hypernyms. As the seed was fixed, this means that the performance on the hypernym-specific validation sets increased if at training time the models saw more examples, from different categories, indicating that a similarity relationship was learned and transferred across different contexts. Improvements over baseline also took place if a metric was learned on datasets where the word pairs did not share a context, as was the case with WS353 and SL999, but the percentual increase was lower, as seen in Table 5.

Comparing the results of BERT contextualized and non-contextualized, the baseline values of the contextualized representations were worse than those obtained with the contextualized embeddings, although without statistical significance, while the improvement after training was consistently better and significant for all datasets with the contextualized representations. Figs. 2c and 2d, show that the distribution of points using the contextualized embeddings is more concentrated and collinear, making it more likely that a metric that acts in the same way for all points in the dataset will rotate and rescale them into a positive correlation. The percentual increases also show that BERT contextualized had the greatest increases from before to after training, suggesting that there was a cumulative effect in considering the context both in the representations and in the similarity measure.

Table 6 shows the results of applying the best model learned on all hypernyms to the WS353 and SL999 datasets. The baseline values for the static representations are comparable with the existing literature (Toshevska et al., 2020). We see that our model was capable of improving on the correlation scores on the datasets, for some representations. Although the improvements did not happen across the board, they show clear evidence that the notion of similarity in the form of a modified cosine measure can be learned in one dataset and applied with positive results to an independent dataset.

		Pearson		Spearman	
		WS353	SL999	WS353	SL999
BERT	Model	**0.487**	**0.375**	**0.519**	**0.384**
	Base	0.239	0.151	0.267	0.172
GPT-2	Model	0.635	**0.507**	0.676	0.513
	Base	0.647	0.504	0.709	0.520
W2V	Model	0.613	0.472	0.632	**0.457**
	Base	0.653	0.460	0.700	0.452
GloVe	Model	**0.593**	**0.431**	0.558	**0.392**
	Base	0.578	0.408	0.578	0.376
SOTA		0.704	0.658	0.828	0.76

Table 6: Best model trained on all hypernyms, tested on SimLex-999 and WordSim-353 datasets. Bold values indicate correlation scores above baseline, and underlining indicates statistical significance. State of the art from Recski et al. (2016); Dobó and Csirik (2020); Speer et al. (2017); Banjade et al. (2015).

5 Conclusion and Outlook

In this paper we tested whether a contextualized notion of cosine similarity could be learned, improving the similarity not only of the results for the datasets where it was learned, but of unrelated similarities. We showed that this metric improved the correlations above baseline, and that, when learned on a contextualized similarity dataset, it had an advantage when compared to one learned on a dataset with unrelated word-pairs. We furthermore showed that this framework has the potential to generalize the notion of similarity to word-pairs it has not seen during training. An important future research line towards interpretability consists in understanding the properties of the metrics that yielded the best results, particularly in identifying the distinctive features of the best metrics, such as their eigensystems. Other further directions include applying these metrics to distributional compositional contractions, including with dependency enhancements (Kogkalidis et al., 2019), testing this framework on larger contextualized datasets and trying out more complex, non-linear, metric forms.

Acknowledgements

All authors would like to thank Juul A. Schoevers for contributions made during the early stages of the project. A.D.C. would like to thank Gijs Wijnholds, Konstantinos Kogkalidis, Michael Moortgat and Henk T.C. Stoof for the many exchanges during this research. This work is supported by the UU Complex Systems Fund, with special thanks to Peter Koeze.

References

Mina Abbaszadeh, S Shahin Mousavi, and Vahid Salari. 2021. Parametrized quantum circuits of synonymous sentences in quantum natural language processing. *arXiv preprint arXiv:2102.02204*.

Sheldon Jay Axler. 1997. *Linear algebra done right*. Springer.

Rajendra Banjade, Nabin Maharjan, Nobal B Niraula, Vasile Rus, and Dipesh Gautam. 2015. Lemon and tea are not similar: Measuring word-to-word similarity by combining different methods. In *International conference on intelligent text processing and computational linguistics*, pages 335–346.

Marco Baroni, Raffaella Bernardi, and Roberto Zamparelli. 2014. Frege in space: A program for composition distributional semantics. In *Linguistic Issues in Language Technology, Volume 9, 2014-Perspectives on Semantic Representations for Textual Inference*.

Lawrence W Barsalou. 1982. Context-independent and context-dependent information in concepts. *Memory & cognition*, 10(1):82–93.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (ACM FAccT '21)*, pages 610–623.

Christopher M Bishop. 2006. *Pattern recognition and machine learning*. Springer.

M Sadrzadeh B Coecke and S Clark. 2010. Mathematical foundations for a compositional distributed model of meaning. *Lambek Festschirft, Linguistic Analysis*, 36(1-4):345–384.

Adriana D Correia, Michael Moortgat, and Henk TC Stoof. 2020. Density matrices with metric for derivational ambiguity. *Journal of Applied Logics*, 2631(5):795.

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 (NAACL '19)*, pages 4171–4186.

András Dobó and János Csirik. 2020. A comprehensive study of the parameters in the creation and comparison of feature vectors in distributional semantic models. *Journal of Quantitative Linguistics*, 27(3):244–271.

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 (WWW '01)*, pages 406–414.

Robert L Goldstone, Douglas L Medin, and Jamin Halberstadt. 1997. Similarity in context. *Memory & Cognition*, 25(2):237–255.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011a. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP '11)*, page 1394–1404.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011b. Experimenting with transitive verbs in a DisCoCat. In *Proceedings of the 2011 Workshop on GEometrical Models of Natural Language Semantics (GEMS '11)*, pages 62–66.

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 (NAACL '19)*, pages 4129–4138.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Konstantinos Kogkalidis, Michael Moortgat, and Tejaswini Deoskar. 2019. Constructive type-logical supertagging with self-attention networks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP '19)*, pages 113–123.

Brian Kulis et al. 2013. Metric learning: A survey. *Foundations and Trends® in Machine Learning*, 5(4):287–364.

Borut Levart. 2011. Triangles on a sphere. Wolfram Demonstrations Project.

Douglas L Medin, Robert L Goldstone, and Dedre Gentner. 1993. Respects for similarity. *Psychological review*, 100(2):254.

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 (NIPS '13)*, pages 3111–3119.

Dmitrijs Milajevs, Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Matthew Purver. 2014. Evaluating neural word representations in tensor-based compositional settings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP '14)*, pages 708–719.

Denis Paperno, Marco Baroni, et al. 2014. A practical and linguistically-motivated approach to compositional distributional semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL '14)*, pages 90–99.

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 '14)*, pages 1532–1543.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Sebastian Raschka. 2015. *Python machine learning*. Packt publishing.

Gábor Recski, Eszter Iklódi, Katalin Pajkossy, and András Kornai. 2016. Measuring semantic similarity of words using concept networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP (RepL4NLP '16)*, pages 193–200.

Russell Richie and Sudeep Bhatia. 2021. Similarity judgment within and across categories: A comprehensive model comparison. *Cognitive Science*, 45(8):e13030.

Russell Richie, Bryan White, Sudeep Bhatia, and Michael C Hout. 2020. The spatial arrangement method of measuring similarity can capture high-dimensional semantic structures. *Behavior research methods*, 52(5):1906–1928.

Gerard Salton and Michael J McGill. 1983. *Introduction to modern information retrieval*. McGraw Hill.

Mykhaylo Schwarz, Mykhaylo Lobur, and Yuriy Stekh. 2017. Analysis of the effectiveness of similarity measures for recommender systems. In *Porceedings of the 14th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM '17)*, pages 275–277.

Grigori Sidorov, Alexander Gelbukh, Helena Gómez-Adorno, and David Pinto. 2014. Soft similarity and soft cosine measure: Similarity of features in vector space model. *Computación y Sistemas*, 18(3):491–504.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI '17)*, pages 4444–4451.

Martina Toshevska, Frosina Stojanovska, and Jovan Kalajdjieski. 2020. Comparative analysis of word embeddings for capturing word similarities. *arXiv preprint arXiv:2005.03812*.

Amos Tversky. 1977. Features of similarity. *Psychological review*, 84(4):327.

Robert M Wald. 2010. *General relativity*. University of Chicago Press.

Gijs Wijnholds and Mehrnoosh Sadrzadeh. 2019. Evaluating composition models for verb phrase elliptical sentence embeddings. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL '19)*, pages 261–271.

Ian H Witten, Eibe Frank, Mark A Hall, and CJ Pal. 2005. *Data Mining: Practical machine learning tools and techniques*. Elsevier.

Context Matters in Semantically Controlled Language Generation for Task-oriented Dialogue Systems

Ye Liu[1], **Wolfgang Maier**[2], **Wolfgang Minker**[3] and **Stefan Ultes**[4]
[1,2,4]Mercedes-Benz AG, Sindelfingen, Germany
[1,3]Ulm University, Ulm, Germany
[1]ye.y.liu@daimler.com
[2,4]{wolfgang.mw.maier,stefan.ultes}@daimler.com
[3]wolfgang.minker@uni-ulm.de

Abstract

This work combines information about the dialogue history encoded by pre-trained models with a meaning representation of the current system utterance to realize contextual language generation in task-oriented dialogues. We utilize the pre-trained multi-context ConveRT model for context representation in a model trained from scratch; and leverage the immediate preceding user utterance for context generation in a model adapted from the pre-trained GPT-2. Both experiments with the MultiWOZ dataset show that contextual information encoded by pre-trained models improves the performance of response generation both in automatic metrics and human evaluation. Our presented contextual generator enables higher variety of generated responses that fit better to the ongoing dialogue. Analysing the context size shows that longer context does not automatically lead to better performance, but the immediate preceding user utterance plays an essential role for contextual generation. In addition, we also propose a re-ranker for the GPT-based generation model. The experiments show that the response selected by the re-ranker has a significant improvement on automatic metrics.

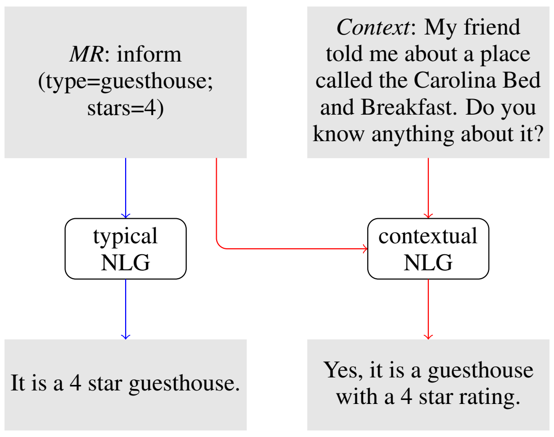

MR: inform (type=guesthouse; stars=4)
Context: My friend told me about a place called the Carolina Bed and Breakfast. Do you know anything about it?
typical NLG
contextual NLG
It is a 4 star guesthouse.
Yes, it is a guesthouse with a 4 star rating.

Figure 1: An example of contextual NLG (red part) compared with typical NLG (blue part) in our experiments. (More examples with multi-turn context please find in the Appendix)

1 Introduction

In a conversation, speakers are influenced by previous utterances and tend to adapt their way of speaking to each other (Dušek and Jurcicek, 2016; Reitter et al., 2006). Furthermore, generating the responses that fit well to dialogue context facilitates successful conversation and strengthens the user's impression of Spoken Dialogue Systems (SDSs). Several previous works (Dušek and Jurcicek, 2016; Kale and Rastogi, 2020; Sankar et al., 2019) have explored the impact of previous dialogue information on the generated language in task-oriented dialogue. However, how to efficiently infuse the dialogue context into a semantically controlled generator for improving the contextual interactive experience is still challenging. Such as, the contextual generator proposed in Dušek and Jurcicek (2016) has no big improvement without the help of an additional re-ranker. The empirical study in Sankar et al. (2019) demonstrated that both recurrent and transformer-based seq2seq model can not effectively consider previous dialogue history for generation. In this work, we propose two contextual generators, which both utilize pre-trained models to encode dialogue context. And the experiment results show that context does matter in semantically controlled task-oriented Natural Language Generation (NLG).

The function of NLG in task-oriented SDS is to generate meaningful output in the form of natural language with the guidance of meaning representation (MR). The MR is a formalism of response semantics and generally represents a dialogue action (DA), such as *inform* or *request*, along with one or more slots and their values (See the MR in

Figure 1). However, this typical NLG only takes the input MR into account and has no clue on how to adapt to the dialogue history. This results in coarse and flat responses (see the blue part in Figure 1). To enable the contextual interaction in SDS, task-oriented NLG should not only contain the desired MR information, but also have access to the dialogue history. The example shown in Figure 1 taken from our experiments: compared with non-contextual generation, the response with context guidance adapts better to the preceding dialogue and is more like a sentence from a real human. In addition, the contextual NLG models are prone to generate more diverse responses.

In summary, the main contributions of this paper are as follows:

- To leverage the contextual nature of the multi-turn dialogue, we utilize the pre-trained multi-context ConveRT (Henderson et al., 2020) to encode dialogue history for the contextual generator. These contextual embeddings are then forwarded to the **S**emantically **C**ontrolled **LSTM** (SC-LSTM) generator (Wen et al., 2015). The **C**onveRT initialized **SC-LSTM** is called *CSC-LSTM* for short in this paper. With the powerful multi-context encoding of ConveRT, we also analyse the impact of variable context size on *CSC-LSTM*. To the best of our knowledge, we are the first to utilize the pre-trained conversational model ConveRT for contextual generation in task-oriented dialogue system.

- We leverage only the immediate preceding user utterance for contextual generation. Adapted from GPT-2 (Radford et al., 2019), the user utterance and DA guide the contextual generation as context and semantic information respectively. We call **C**ontextually and **S**emantically **C**onditioned **GPT** *CSC-GPT* for short in this paper. The experiments of GPT-based contextual model show that generation benefits from dialogue context, even if only immediate preceding user utterance is taken into account.

- We propose a **BERT**-based (Devlin et al., 2019) **R**e-ranker (*BERT-R*) for the *CSC-GPT* generator, to select system response that fit better to the user utterance. Given the top 5 generations of *CSC-GPT*, several automatic scores are regarded as contextual basis between user utterance and system response to train on a multiple regression task adapted from BERT. Experiments show that the re-selected generation has a significant improvement on the performance scores.

The remainder of this paper is structured as follows: Section 2 shows the related works of our research. Section 3 introduces the dataset and the automatic metric scores which are used in this work. Section 4 describes our proposed three models: *CSC-LSTM*, *CSC-GPT* and *BERT-R*, as well as the experiment details. Section 5 shows the experiment results of all models in automatic metrics and human evaluation. The last Section 6 concludes and outlines future research.

2 Related Works

For task-oriented NLG, semantically controlled neural models play a significant role. Wen et al. (2015) introduced a semantically conditioned model by adding an additional semantic cell in LSTM to control the DA, which is defined as the combination of intent and slot-value pairs, for generation. Tseng et al. (2018) improved the RNN-based generator by considering latent information using the semantically conditional variational autoencoder architecture. As the major advantage and superior performance of pre-trained LMs (Devlin et al., 2019; Radford et al., 2019), Peng et al. (2020b) proposed a semantically controlled generation model based on GPT-2; Chen et al. (2020) and Peng et al. (2020a) presented an end-to-end task-oriented SDS based on the pre-trained GPT-2. Even though there has been plenty of works on semantically guided NLG, most approaches fail in utilizing information of the preceding interaction.

Dušek and Jurcicek (2016) stood out as they extended the idea of NLG from MRs by adding *one* preceding user utterance to their recurrent encoder. However, we are more interested in the influence of bigger context sizes in *CSC-LSTM* contextual generation. And the model in Dušek and Jurcicek (2016) was not able to show any improvement for contextual generation without an additional n-gram match re-ranker. However, our proposed contextual generators outperform both the baselines even without re-ranker and the re-ranker in our work can further highly improve the generation performance on automatic metrics. Sankar et al. (2019) made an empirical study to understand how models use the available dialog history for generation and found that both recurrent and transformer-based seq2seq model can not consider previous dialogue history effectively. However, the dialogue history in our work is encoded by pre-trained models and experiments show that the generation in task-oriented dialogue benefits from dialogue history. Kale and Rastogi (2020) also examined the role of context and demonstrated that the generation benefits from the dialogue history. While their approach highly relies on manually pre-defined templates which are costly to create, in this work, all responses are directly generated without the need of any templates.

3 Dataset and Automatic Metrics

In our work, the automatic metric scores are not only used for performance evaluation, but also used for the *BERT-R* training in Section 4.3. Hence, we introduce the dataset and the automatic metrics beforehand in this section.

3.1 MultiWOZ dataset

The original MultiWOZ (Budzianowski et al., 2018) dataset is a fully-labeled collection of human-human written conversations spanning over multiple domains and topics. It contains over $10,000$ dialogues spanning 8 domains, namely: Restaurant, Hotel, Attraction, Taxi, Train, Hospital, Bus, and Police. The test and validation sets contain $1,000$ examples each for performance comparison. The MultiWOZ 2.1 (Eric et al., 2020) and MultiWOZ 2.2 (Zang et al., 2020) both fix some dialogue state annotation errors and dialogue utterances, resulting in an improved version of the original MultiWOZ.

The MultiWOZ 2.1 is used for evaluation of *CSC-LSTM* in Section 4.1 in accordance with related work. The MultiWOZ 2.1 and 2.2 are both used additionally for evaluating *CSC-GPT* and *BERT-R* in Section 4.2 and 4.3.

3.2 Automatic metrics

The following metrics are used for performance comparison in Section 5 and several are applied for *BERT-R* training as target score in Section 4.3.

3.2.1 *N*-gram matching metrics

BLEU-4 is the 4-gram BLEU score (Papineni et al., 2002), which is the most widely used metric score for evaluating the performance of language generation and machine translation. In this work, BLEU-4 is computed for multiple values of $n = (1, 2, 3, 4)$ with weights $(0.25, 0.25, 0.25, 0.25)$ respectively and the scores are averaged geometrically. A smoothing function is used to avoid that no n-gram overlaps are found.

The target signal $\tau_{\text{BLEU-4}}$ in Section 4.3 represents the BLEU-4 score between system generated response and gold reference in *BERT-R*.

Meteor (Banerjee and Lavie, 2005) is a kind of weighted F-score based on mapping unigrams and also computes a penalty function for incorrect word order. Lavie and Agarwal (2007) demonstrated that Meteor score has high correlation with human ratings.

The target signal τ_{Meteor} in Section 4.3 represents the Meteor score between system generation and gold reference in *BERT-R*.

3.2.2 Machine learned metric

BERTScore (Zhang et al., 2019) is a machine learned automatic evaluation metric for text generation that has shown a high correlation with human judgments. BERTScore leverages the pre-trained contextual embeddings from variants of BERT (Devlin et al., 2019) and matches words in candidate and reference sentences by cosine similarity. Moreover, BERTScore computes precision, recall, and F1 measure[1]. Zhang et al. (2019) showed that the Roberta (Liu et al., 2019) large model has the best-performing results for English tasks. So the roberta-large model[2] is used for computing BERTScore in this work.

The target signals $\tau_{\text{BERT}_{\text{pre}}}$, $\tau_{\text{BERT}_{\text{rec}}}$, $\tau_{\text{BERT}_{\text{f1}}}$ in Section 4.3 represent the precision, recall and F1 of BERTScore between system response and gold reference respectively in *BERT-R*.

3.2.3 Other metrics:

ConveRT cosine similarity: ConveRT (Henderson et al., 2020) is a light-weight conversational model pre-trained on the large Reddit conversational corpus (Henderson et al., 2019). It provides powerful representations for conversational data and can be used as a response ranker by comparing the cosine similarity between user utterance and multiple responses. In this work, we not only utilize the pre-trained ConveRT for context embedding in *CSC-LSTM*, but also for a target score in *BERT-R*.

The target signal $\tau_{\text{ConveRT-cs}}$ in Section 4.3 means

[1] Only F1 score, which represents a reasonable balance between recall and precision, is shown in Table 1, Table 2, Table 4, Table 5 and Figure 2 for performance comparison.

[2] https://github.com/Tiiiger/bert_score

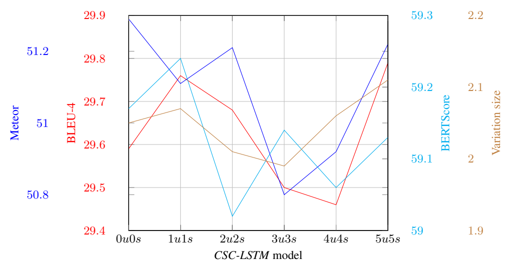


Figure 2: The curves of BLEU-4, Meteor, BERTScore and Variation size over CSC-LSTM model with different context size (the exact value of metrics please find in Table 1) show similar tendency: the both ends of every curve have better performance than the inner part.

the cosine similarity of ConveRT embedding between user utterance and system response in *BERT-R*.

Variation size measures the variation of the generated system responses, i.e., how many different realisations are generated for one DA on average. The variation size results computed on the full MultiWOZ test data are shown in the left part of the results column (/) while the right part shows the variation size computed only over the instances of the test data with DA that appear more than once, i.e., where variation can actually occur.

4 The Proposed Models and Experiment Details

In this section, our proposed models (two contextual generators, one response re-ranker) and corresponding experiment details are introduced.

4.1 ConveRT initialized SC-LSTM: *CSC-LSTM*

We train *CSC-LSTM*[3] on the basis of the SC-LSTM (Wen et al., 2015), where a semantic control cell encodes DA into an one-hot embedding to guide the task-oriented generation that is oblivious about any dialogue history. In our proposed *CSC-LSTM*, we apply the pre-trained multi-context ConveRT[4] for encoding the dialogue history and the contextual embedding is forwarded to initialize the SC-LSTM generator. Before initialization in *CSC-LSTM*, a non-linear transformation[5] is applied, which is shown in Eq. 1, to project the ConveRT embedding into the SC-LSTM decoder space:

$$h_0 = tanh(\mathbf{W} C_e + \mathbf{b}) \ . \qquad (1)$$

$h_0 \in \mathbb{R}^{d_g}$ is the SC-LSTM decoder initial recurrent state, $C_e \in \mathbb{R}^{d_c}$ is the ConveRT context embedding and $\mathbf{W} \in \mathbb{R}^{d_g \times d_c}$ projects the context level embedding into the decoder space. The $\mathbf{W}$ and $\mathbf{b}$ are learnable parameters during the *CSC-LSTM* training.

Experiment details of *CSC-LSTM*: For *CSC-LSTM*, the SC-LSTM will be used as baseline without additional context information. This means, for each utterance generation, the hidden state is initialized with zeros in SC-LSTM. The MultiWOZ 2.1 (Eric et al., 2020) dataset is used for SC-LSTM and *CSC-LSTM* generation models.

The multi-context ConveRT embedding dimensionality is 512, hence, the d_c in Eq. 1 is 512 for *CSC-LSTM* training. In order to ensure a fair comparison, we set the same hyper-parameters for SC-LSTM and *CSC-LSTM*: the hidden size to 300 (the d_g in Eq. 1), the learning rate to $5e{-}3$, the batch size to 128 and beam search decoding in inference with beam size 10. Early stopping and cross entropy loss are applied during the SC-LSTM and *CSC-LSTM* training. The responses in SC-LSTM and *CSC-LSTM* are delexicalised text where the slot values are replaced by its corresponding slot tokens.

Context size analysis of *CSC-LSTM*: The pre-trained multi-context ConveRT does not only encode the immediate preceding user utterance but in addition a maximum of 10 previous dialogue sentences, i.e., 5 user utterances and 5 system responses ($5u5s$). To analyse the effect of this context on the performance of the *CSC-LSTM*, multiple models with different context sizes have been trained. And we plot the trend and show exact values of all metric scores in Figure 2 and Table 1 respectively. The $0u0s$ in Figure 2 and Table 1 means only immediate preceding user utterance without extra context is taken into account for contextual generator *CSC-LSTM* training.

4.2 Contextually and Semantically Conditioned GPT: *CSC-GPT*

In addition to the contextual generator trained from scratch in Section 4.1, we also explore contextual generation adapted from a pre-trained LM model in this section.

[3]The architecture of *CSC-LSTM* is shown in Appendix.
[4]https://github.com/davidalami/ConveRT
[5]Using the same hidden size as the dimension of the ConveRT embedding, i.e., *CSC-LSTM* (hidden size 512) without project function d_0 in Eq. 1 results in worse performance.

context size	*0u0s*	*1u1s*	*2u2s*	*3u3s*	*4u4s*	*5u5s*
BLEU-4 (%)	29.59	29.76	29.68	29.50	<u>29.46</u>	**29.79**
Meteor (%)	**51.29**	51.11	51.21	<u>50.80</u>	50.92	51.22
BERTScore F1 (%)	59.17	**59.24**	<u>59.02</u>	59.14	59.06	59.13
Variation size	2.05	2.07	2.01	<u>1.99</u>	2.06	**2.11**

Table 1: The results for BLEU, Meteor, BERTScore and Variation size of *CSC-LSTM* model with variant context size (best results are marked with bold font and worst results are marked with underline) show that the best models exist in *0u0s*, *1u1s* and *5u5s*, while the worst models exist in *2u2s*, *3u3s* and *4u4s*.

We train *CSC-GPT*[6] on the basis of the pre-trained GPT-2 (Radford et al., 2019). It adopts the generic Transformers (Vaswani et al., 2017). Peng et al. (2020b) already proposed the SC-GPT model, which was continuously training the GPT-2 on (DA, system response) pairs. However, no context information was taken into account in SC-GPT for dialogue response generation. In our proposed *CSC-GPT*, we leverage the extra user context beyond semantic information to guide the generation process. This means, (user utterance, DA, system response) MultiWOZ triplets are continuously trained on the pre-trained GPT-2 model for contextual generation. Given the extra context size analysis result of *CSC-LSTM* model (shown in 5.1) and GPU memory limitation for training the pre-trained LM, only the immediate preceding user utterance is used as context information in the *CSC-GPT*[7].

In this work, we tackle the generation problem using conditional LM. Given the dialogue dataset $\mathcal{D} = \{(u_n, d_n, r_n)\}_{n=1}^{N}$ with N samples, the goal is to build a statistical model parameterized by θ to characterize $p_\theta(r|u,d)$, which can be written as the product of a series of conditional probabilities.

$$p_\theta(r|u,d) = \prod_{t=1}^{T} p_\theta(r_t|r_{<t}, u, d) \tag{2}$$

where $r_{<t}$ indicates all tokens before t. The u represents user utterance; d means the system DA and r is the system response which includes $(r_1, r_2, ...r_t, ...)$ tokens with length T.

Experiment details of *CSC-GPT*: In order to achieve a robust performance comparison, two datasets, namely MultiWOZ 2.1 and MultiWOZ 2.2, are used in SC-GPT and *CSC-GPT*. During training, the batch size is 16, the maximal epoch is 10, the learning rate is 5e−5 and early stopping is used. During decoding, we use the top-k (Fan et al., 2018) and nucleus sampling (top-p) (Holtzman et al., 2019) decoding algorithms with top-k equal to 5 and top-p equal to 0.9. This means, the next token distribution is filtered to keep maximal top 5 tokens with highest probability and the cumulative probability above a 0.9 threshold. Due to the computational expense of running large SC-GPT and *CSC-GPT* model, only the top 5 responses are generated.

4.3 BERT Re-ranker: *BERT-R*

In this paper, we propose a **BERT** (Devlin et al., 2019) **R**e-ranker (*BERT-R*) to select the top generation which is more similar to human sentence and better fits to user context. As the generated responses of the *CSC-LSTM* are delexicalized and have less variability compared with the *CSC-GPT*, we only apply the re-ranker to the *CSC-GPT* model. Adapted from pre-trained BERT, the *BERT-R* is continually trained with task-oriented dialogue data and then fine-tuned on a multiple regression task, where the model learns the relationship between user utterance and system response from the various regression targets defined by multiple metrics scores.

There are two steps in our proposed *BERT-R*: masked LM pre-training and multiple regression fine-tuning. And the BERT-base-uncased[8] model with 12 layers, 768 hidden units and 12 heads is used in this work.

Masked LM Pre-training The original BERT was pre-trained with the BooksCorpus (Zhu et al., 2015) and English Wikipedia. In order to better generalize to task-oriented dialogues, we continually train the BERT model with a dialogue dataset: DSTC8 (Rastogi et al., 2020), which is a schema-guided dialogue dataset and consists of over 20k

[6]The architecture of *CSC-GPT* is shown in Figure 4 in Appendix

[7]In the Appendix, we also conduct the *CSC-GPT* with different context size: *CSC-GPT (0u0s)* and *CSC-GPT (1u1s)*. The performance comparison in Table 4 supports the extra context size analysis in *CSC-LSTM*: longer dialogue context can not linearly improve the generation performance.

[8]https://huggingface.co/bert-base-uncased

annotated multi-domain, task-oriented conversations between a human and a virtual assistant. And Sellam et al. (2020) and Peng et al. (2020b) both continually trained with task-specific data based on the pre-trained LMs for better generalisation. Similar to the masked LM training of the original BERT, only 15% tokens are randomly masked for prediction with cross entropy loss.

Multiple Regression Fine-tuning The ideal generated response should be close to human communication and relevant to preceding user utterance at the same time. The general idea for *BERT-R* is straightforward: using multiple widely used metric scores to guide the model to learn the relationship of user utterance and system response by a multiple regression task, where the regression targets are those metric scores. Those multiple metric scores define how good the generation is from different perspectives to avoid dominance by one single score. Hence, we employ two n-gram matching metrics: BLEU-4 and Meteor score, and a machine learned score: BERTScore, to define how similar system response is with respect to gold reference; and ConveRT cosine similarity to define how contextual system response is with respect to user utterance.

In this work, we continually train *BERT-R* with the guidance of multiple metric scores. Define the user utterance $u = (u_1, ..., u_l)$ of length l where each u_i is a token and system response $r = (r_1, ..., r_m)$ of length m. Let $\mathcal{D}' = \{(u_n, r_n, y_n^{\tau})\}_{n=1}^{N'}$ be a training dataset of size N', where τ is a target signal. y^{τ} represents all metric scores: $y^{\tau} = (y^{\tau_{\text{BLEU-4}}}, y^{\tau_{\text{Meteor}}}, y^{\tau_{\text{BERT}_{\text{pre}}}}, y^{\tau_{\text{BERT}_{\text{rec}}}}, y^{\tau_{\text{BERT}_{\text{f1}}}}, y^{\tau_{\text{ConveRT-cs}}})$. Hence, the size of τ is 6, which means 6 specific regression layers will be added to the output of [CLS] token in *BERT-R*. y^{τ} will guide *BERT-R* to learn how similar system response r is with respect to gold reference and how contextual system response r is with respect to user utterance u during the fine-tuning. Given the training data, the goal of fine-tuning is to learn a multiple regression function $f : (u, r) \rightarrow y^{\tau}$ that predicts different metric scores.

Given the sentence pair (u, r), the pre-trained *BERT-R* returns a sequence of contextualized vectors:

$$v_{[\text{CLS}]}, v_{u_1}, ..., v_{u_l}, v_{r_1}, ..., v_{r_m} = \textit{BERT-R}(u, r) \tag{3}$$

where $v_{[\text{CLS}]}$ is the *BERT-R* output representation for the special [CLS] token, which can be further fine-tuned for classification or regression task. As described by Devlin et al. (2019), we add separate linear layers on top of the [CLS] vector to predict different metric scores:

$$\hat{y}^{\tau} = f(u, r) = \mathbf{W}^{\tau} v_{[\text{CLS}]} + \mathbf{b}^{\tau} \tag{4}$$

where $\mathbf{W}^{\tau}$ and $\mathbf{b}^{\tau}$ are the weight matrix and bias vector respectively. And we use the Eq. 5, the sum of all target-specific regression loss to fine-tune *BERT-R*.

$$l_{\text{fine-tuning}} = \sum_{\tau} \left(\frac{1}{N'} \sum_{n=1}^{N'} ||\hat{y}_n^{\tau} - y_n^{\tau}||^2 \right) \tag{5}$$

After fine-tuning, the *BERT-R* is used to select the top generation with the highest score: the sum of all regression output of *BERT-R*, which is shown in Eq. 6:

$$S_{\textit{BERT-R}} = \sum_{\tau} \textit{BERT-R}(u, r) \tag{6}$$

Experiment details of *BERT-R*: During masked LM pre-training, batch size is 32, maximal epoch is 10 and learning rate is 5e−5. And early stopping is used to avoid over-fitting on the DSTC8 training dataset.

For the fine-tuning of *BERT-R*, we generate the top 5 responses for MultiWOZ data with *CSC-GPT* model firstly. And in order to clean the system generated responses, we have the following procedures:

1) The duplicated system responses are removed.

2) The last turn of every dialogue is removed, where there are always “thank you” and “good bye”, kind of non-informative sentences.

3) In order to let the network glance what the human communication looks like, we add the gold reference for the user utterance in the training dataset. In addition, we need to remove the system responses which are same as the gold reference beforehand, to comply with the rule 1).

After finishing the above process, we compute the target score respectively for the label of different regression layers. During the training of the multiple regression task, the batch size is set to 32, learning rate 1e−5, and early stopping is used to save the best BERT re-ranker.

	MultiWOZ 2.1				MultiWOZ 2.2			
	BLEU-4 (%)	Meteor (%)	BERTScore F1 (%)	Variation size	BLEU-4 (%)	Meteor (%)	BERTScore F1 (%)	Variation size
SC-LSTM (Wen et al., 2015)	28.76	49.93	58.71	1.00/1.00	-	-	-	-
CSC-LSTM ($5u5s$)	**29.79**	**51.22**	**59.13**	**1.35/2.11**	-	-	-	-
SC-GPT (Peng et al., 2020b)	28.95	50.22	91.96	2.27/6.56	28.53	49.80	91.95	2.78/6.70
CSC-GPT	29.91	51.34	92.08	**2.29/6.66**	29.41	51.10	92.08	**2.81/6.82**
BERT-R	**32.37**	**54.01**	**92.40**	2.22/6.34	**31.68**	**53.65**	**92.39**	**2.81/6.82**

Table 2: The results for BLEU, Meteor, BERTScore and variation size of top one generation in all models demonstrate that our proposed contextual models in: *CSC-LSTM* against SC-LSTM and *CSC-GPT* against SC-GPT, both outperform the corresponding baseline. Meanwhile, the proposed re-ranker *BERT-R* also highly improve the metric scores compared with all other models.

5 The Experiment Results

The experimental results of all models introduced in Section 4 are presented in this section. To ensure a consistent performance comparison, we compute the metric scores based on the top one generation of all models. Table 2 shows the results of all automatic metrics.

Furthermore, a **human evaluation** has been conducted. We randomly sampled 100 dialogues and their corresponding top one generations from our proposed models as well as the baselines. We recruited three annotators with relevant background in SDS to evaluate the responses generated by different models. Each rater was presented the complete preceding dialogue and asked to rate if "The highlighted system response could plausibly have been produced by a human" (natural) and if "The highlighted system response fits well to the previous dialog" (contextual). Each metric is rated on a 5-point Likert scale, where 1 is "not agree at all", 5 is "fully agree". In order to guarantee the strictness of human evaluation, the human judges have no information about the origin of the utterances, i.e., which model generated the utterance. Table 3 shows the human evaluation results.

	Contextual	Natural
SC-LSTM (Wen et al., 2015)	3.96	4.04
CSC-LSTM ($5u5s$)	**4.21***	**4.16***
SC-GPT (Peng et al., 2020b)	4.00	4.14
CSC-GPT	**4.25**$^+$	**4.27**$^+$
BERT-R	4.18	4.26

Table 3: The results of human evaluation on natural and contextual score of all models. ($*$: p-value < 0.01, comparison with SC-LSTM baseline; $+$: p-value < 0.1, comparison with SC-GPT baseline) show the superiority of our proposed contextual models.

5.1 Experiment results of *CSC-LSTM*

The automatic metric scores comparing SC-LSTM and *CSC-LSTM* in Table 2 show that *CSC-LSTM* has the overall better performance in BLEU-4, Meteor, BERTScore and variation size compared to the baseline. The variation size results show *CSC-LSTM* can generate more variant responses per DA, which may indicate a more contextual fitting response, while the SC-LSTM only generate the same utterances each time. The performance comparison between SC-LSTM and *CSC-LSTM* in Table 2 support our initial assumption that context helps to generate good system utterances. Especially the increase in variation size is of importance as it indicates that the resulting utterances of *CSC-LSTM* indeed be different for different contexts.

This has been validated by the human evaluation of SC-LSTM and *CSC-LSTM* in Table 3. It shows that the variation introduced by *CSC-LSTM* actually results in utterances that fit significantly better to the preceding dialogue and are perceived as significantly more natural. Overall, the performance comparison between SC-LSTM and *CSC-LSTM* on automatic metrics and human evaluation demonstrate the dialogue history contributes to contextual and variant responses.

The context size analysis of *CSC-LSTM* demonstrate that the automatic metric scores are influenced by the length of the context. All metrics show similar curves over the different contextual model in Figure 2. The both ends of the curves have better performance than the inner part. The Table 1 shows the best BLEU-4 and variation size are both achieved for context sizes of $5u5s$; while $0u0s$ has the best Meteor and $1u1s$ has the best BERTScore. Both show that the contextual models $0u0s$, $1u1s$ and $5u5s$ generally outperform the

$2u2s$, $3u3s$ and $4u4s$. Hence, the investigation of the impact of context size for *CSC-LSTM* generation indicates that longer context does not linearly result in better performance, which is further confirmed in the performance comparison between *CSC-GPT*$(0u0s)$ and *CSC-GPT*$(1u1s)$ shown in Table 4 in Appendix. Evidently, all contextual models achieve better performance than the baseline (SC-LSTM in Table 2). We therefore conclude that the immediate preceding user utterance yields the indispensable information for contextual generation. With limited memory, using only immediate preceding user utterance without extra context can be regarded as a balanced option that we directly apply for training the *CSC-GPT*.

5.2 Experiment results of *CSC-GPT*

All scores between SC-GPT and *CSC-GPT* in Table 2 demonstrate that the *CSC-GPT* is superior to the baseline SC-GPT for both datasets, MultiWOZ 2.1 and MultiWOZ 2.2. Our assumption is again confirmed in GPT-based generation model: dialogue history contributes to contextual and variant response, even though only one preceding user utterance is taken into account.

When comparing the human evaluation results of SC-GPT and *CSC-GPT* in Table 3, the assumption is further supported: adding context to the generation process results in more natural and contextual responses. The *CSC-GPT* generator even achieves the best rating both in terms of natural and contextual score compared to all other models. This means that enhancing an already powerful pre-trained model with context is essential for its application within dialogue systems.

5.3 Experiment results of *BERT-R*

Our proposed *BERT-R* selects the top one response from 5 *CSC-GPT* generations with the highest score in Eq. 6. The metric scores of *CSC-GPT* and *BERT-R* in Table 2 show that the selected generation by *BERT-R* has a significant improvement on BLEU-4, Meteor and BERTScore[9], with a little loss on variation size compared to the top one generation in *CSC-GPT*. This is in line with human evaluation results where the *CSC-GPT* achieves slightly better scores both in naturalness and contextualness than *BERT-R*, even though *BERT-R* clearly shows better results compared to the SC-GPT baseline.

[9]also improvement on ConveRT cosine similarity, which is shown in the Appendix.

6 Conclusion and Future Work

In this paper, we propose two contextual generation models: *CSC-LSTM* trained from scratch and *CSC-GPT* adapted from pre-trained GPT-2. Both integrate dialogue context information into NLG for generating more variant and contextual response in task-oriented dialogue systems.

In the experiment of *CSC-LSTM* against SC-LSTM and *CSC-GPT* against SC-GPT, our proposed contextual models both improve the generation performance in automatic metrics, thus showing that *CSC-LSTM* and *CSC-GPT* are able to capture better the contextual needs resulting in a higher similarity to the data. This is further underpinned by the number of variations. More variant responses are generated per DA in *CSC-LSTM*, while the SC-LSTM only generates the same utterances each time. Furthermore, the variation size of GPT-based generators is higher than SC-LSTM based models. The possible reason is the pre-trained GPT-2 contributes to more diverse responses by default. The human evaluation results in Table 3 not only demonstrate the contextual model can generate more contextual and natural response compared with their baseline respectively, but also show GPT-2 contextual model *CSC-GPT* is superior than *CSC-LSTM*.

An investigation of the impact of context size for dialogue response generation in *CSC-LSTM* indicates that longer context does not automatically result in better performance. However, all variant *CSC-LSTM* models have better performance than baseline, which means the immediate preceding user utterance contains the most contextual information for generation. This is also verified in GPT-2 contextual generators, even only immediate preceding user utterance is taken into account, the *CSC-GPT* model outperforms SC-GPT both on automatic metrics and human evaluation.

In addition to the above mentioned two contextual models, we also present a re-ranker for *CSC-GPT* contextual model. Adapted from pre-trained BERT, the *BERT-R* continually train on multi-domain dialogues and fine-tune on a multiple regression task to learn the relationship between user context and system response by the metric guidance of BLEU-4, Meteor, BERTScore and ConveRT cosine similarity. Finally, the top one generation selected by *BERT-R* has significant superiority in BLEU-4, Meteor, BERTScore and ConveRT cosine similarity compared with top one generation in *CSC-GPT*. This means, that our proposed *BERT-R* works from the guidance of metric scores and can choose the generation with highest score. However, *CSC-GPT* slightly outperforms *BERT-R* in variation size and human evaluation. The possible reason is that the existing automatic metrics still have bias with human judgments (Chaganty et al., 2018).

In the future, we will further explore the performance of *BERT-R* with the guidance of other automatic metrics, which have higher correlation with human judgements. Furthermore, there seems to be a link between the variation size metric and the human evaluation scores, which will also be part of future work.

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.

Pawel Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *EMNLP*.

Arun Tejasvi Chaganty, Stephen Mussmann, and Percy Liang. 2018. The price of debiasing automatic metrics in natural language evalaution. In *ACL (1)*.

Zhiyu Chen, Harini Eavani, Wenhu Chen, Yinyin Liu, and William Yang Wang. 2020. Few-shot nlg with pre-trained language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 183–190.

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.

Ondřej Dušek and Filip Jurcicek. 2016. A context-aware natural language generator for dialogue systems. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 185–190.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. Multiwoz 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 422–428.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *ACL (1)*.

Matthew Henderson, Paweł Budzianowski, Iñigo Casanueva, Sam Coope, Daniela Gerz, Girish Kumar, Nikola Mrkšić, Georgios Spithourakis, Pei-Hao Su, Ivan Vulić, et al. 2019. A repository of conversational datasets. In *Proceedings of the First Workshop on NLP for Conversational AI*, pages 1–10.

Matthew Henderson, Iñigo Casanueva, Nikola Mrkšić, Pei-Hao Su, Tsung-Hsien Wen, and Ivan Vulić. 2020. Convert: Efficient and accurate conversational representations from transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 2161–2174.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Mihir Kale and Abhinav Rastogi. 2020. Template guided text generation for task oriented dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6505–6520.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the second workshop on statistical machine translation*, pages 228–231.

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*.

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.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2020a. Soloist: Few-shot task-oriented dialog with a single pretrained auto-regressive model. *arXiv preprint arXiv:2005.05298*.

Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020b. Few-shot natural language generation for task-oriented dialog. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 172–182.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8689–8696.

David Reitter, Frank Keller, and Johanna D Moore. 2006. Computational modelling of structural priming in dialogue. In *Proceedings of the human language technology conference of the naacl, companion volume: Short papers*, pages 121–124.

Chinnadhurai Sankar, Sandeep Subramanian, Chris Pal, Sarath Chandar, and Yoshua Bengio. 2019. Do neural dialog systems use the conversation history effectively? an empirical study. In *ACL (1)*.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. Bleurt: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892.

Bo-Hsiang Tseng, Florian Kreyssig, Paweł Budzianowski, Iñigo Casanueva, Yen-Chen Wu, Stefan Ultes, and Milica Gasic. 2018. Variational cross-domain natural language generation for spoken dialogue systems. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 338–343.

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

TH Wen, M Gašić, N Mrkšić, PH Su, D Vandyke, and S Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Conference Proceedings-EMNLP 2015: Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721.

Xiaoxue Zang, Abhinav Rastogi, and Jindong Chen. 2020. Multiwoz 2.2: A dialogue dataset with additional annotation corrections and state tracking baselines. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 109–117.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

A Appendices

In order to better understand what our proposed architectures look like, please find the following Figure 3 and Figure 4. The model *CSC-GPT* generates contextual response with the guidance of immediate preceding user utterance and DA. In the *CSC-LSTM*, the SC-LSTM was proposed in Wen et al. (2015) and an additional cell was introduced into the LSTM cell to gate the DA information. The original LSTM cell follows:

$$\begin{aligned}
i_t &= \sigma(\mathbf{W}_{wi}w_t + \mathbf{W}_{hi}h_{t-1}) \\
f_t &= \sigma(\mathbf{W}_{wf}w_t + \mathbf{W}_{hf}h_{t-1}) \\
o_t &= \sigma(\mathbf{W}_{wo}w_t + \mathbf{W}_{ho}h_{t-1}) \\
\hat{c}_t &= \tanh(\mathbf{W}_{wc}w_t + \mathbf{W}_{hc}h_{t-1}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \hat{c}_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned} \tag{7}$$

In SC-LSTM, the d_0 starts from an one-hot vector, at each time step the DA cell decides what information should be retained for future time steps and discards the others. Like:

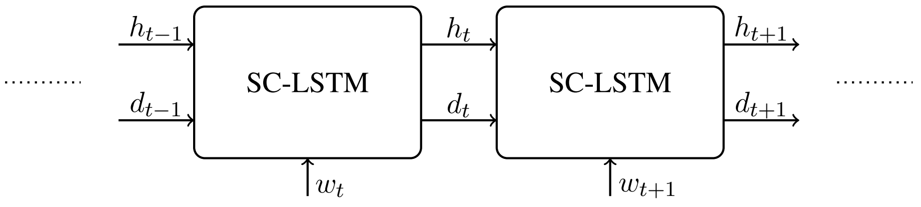
$$\begin{aligned}
r_t &= \sigma(\mathbf{W}_{wr}w_t + \alpha(\mathbf{W}_{hr}h_{t-1})) \\
d_t &= r_t \cdot d_{t-1}
\end{aligned} \tag{8}$$

Then, the value cell in Eq. 7 also depends on the DA,

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t + \tanh(\mathbf{W}_{dc}d_t) \tag{9}$$

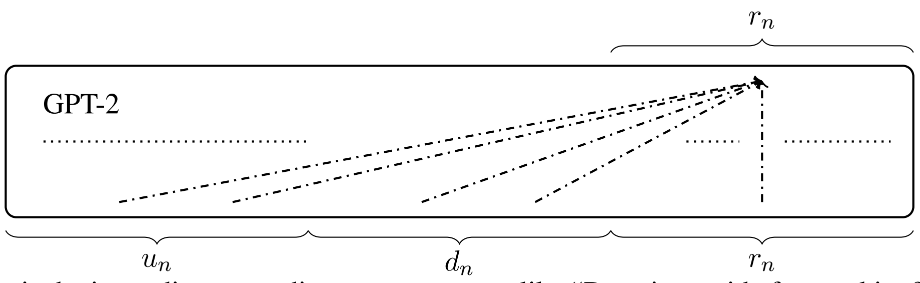
Finally, the hidden state is further updated by new value cell. In our proposed *CSC-LSTM*, not only DA cell is added, but the SC-LSTM cell is initialized by contextual ConveRT embedding.

Given the pre-trained model can handle the longer dialogue context for generation, so we also trained *CSC-GPT* contextual model with one more turn context besides the immediate preceding user utterance, i.e. $1u1s$. And the results comparison between *CSC-GPT (0u0s)* and *CSC-GPT (1u1s)* is shown in Table 4. We can know that the BLEU-4 and Meteor of *CSC-GPT (0u0s)* outperforms *CSC-GPT (1u1s)*, meanwhile, the BERTScore and variation size have no big difference. This further demonstrate that longer context can not directly result in better performance. The maximal length of input in *CSC-GPT (0u0s)* is 120, however, *CSC-GPT (1u1s)* is 190 with more GPU memory. Hence,



h_0 refers to Eq. 1, dialogue context is encoded by ConveRT and projected to decoder space with the Eq. 1
d_0 is DA one-hot embedding, like (0, 0, ..., 1, ...) represents "hotel{ request(area=?)}"
w_t is the input token at time slot t

Figure 3: The architecture of *CSC-LSTM*.



u_n is the immediate preceding user utterance, like "Does it provide free parking?"
d_n is the DA, like "hotel {inform (internet=yes; parking=yes)}"
r_n is the gold response, like "Yes, free parking and free wifi!"

Figure 4: The architecture of *CSC-GPT*. The immediate preceding user utterance and MR together guide the contextual response generation.

we recommend only immediate preceding user utterance is taken into account for contextual generation in task-oriented dialogue system.

The *BERT-R* is trained with multiple metric scores: BLEU-4, Meteor, BERTScore and ConveRT cosine similarity. In order to make consistent comparison of all models, we don't show the ConveRT cosine similarity of *BERT-R* in the main paper. The Table 5 shows that all scores get improvement.

There are several use cases in Table 6. And by system response comparison of SC-LSTM agaist *CSC-LSTM* and SC-GPT agaist *CSC-GPT* given preceding dialogue context, the Table 6 shows the importance of dialogue context for natural and contextual response.

	MultiWOZ 2.1				MultiWOZ 2.2			
	BLEU-4 (%)	Meteor (%)	BERTScore F1 (%)	Variation size	BLEU-4 (%)	Meteor (%)	BERTScore F1 (%)	Variation size
CSC-GPT (0u0s)	**29.91**	**51.34**	**92.08**	**2.29**/6.66	**29.41**	**51.10**	**92.08**	**2.81/6.82**
CSC-GPT (1u1s)	29.70	51.02	**92.08**	**2.29/6.67**	28.80	50.32	91.94	**2.81**/6.81

Table 4: The results for BLEU, Meteor, BERTScore and Variation size of top one *CSC-GPT (0u0s)* generation and *CSC-GPT (1u1s)* show that *CSC-GPT (0u0s)* outperforms *CSC-GPT (1u1s)* and support the context size analysis of *CSC-LSTM*: the longer context in contextual model can not linearly result in better performance.

	MultiWOZ 2.1				MultiWOZ 2.2			
	BLEU-4 (%)	Meteor (%)	BERTScore F1 (%)	ConveRT cs	BLEU-4 (%)	Meteor (%)	BERTScore F1 (%)	ConveRT cs
CSC-GPT	29.91	51.34	92.08	33.41	29.41	51.10	92.08	33.81
BERT-R	**32.37**	**54.01**	**92.40**	**34.70**	**31.68**	**53.65**	**92.39**	**35.00**

Table 5: The results for BLEU, Meteor, BERTScore and ConveRT cosine similarity of top one *CSC-GPT* generation and the selected one by *BERT-R* demonstrate that all target metrics used in *BERT-R* have a significant improvement through re-ranking.

context	**User:** Hi, I'm looking for a hotel called aylesbray lodge guest house. Do you have anything with that name?
system response	SC-LSTM: It is a moderately priced hotel in the south. Would you like me to book it for you?
	CSC-LSTM: Yes, it is a hotel in the south in the moderately priced price range. Would you like to book a room?
	SC-GPT: It is a moderately priced hotel in the south area. Would you like me to book it for you?
	CSC-GPT: I do, it is a moderately priced hotel in the south. Would you like to make a booking?
context	**User:** I'm looking for a nightclub to go to. Are there any in the centre of town?
	System: There are 5 nightclubs in the centre. Ballare is one of them. Would that work for you?
	User: Is there an entrance fee for Ballare?
system response	SC-LSTM: The entrance fee is 5 pounds.
	CSC-LSTM Yes, it is 5 pounds .
	SC-GPT: The entrance fee is 5 pounds.
	CSC-GPT: Yes, it is 5 pounds to get in.
context	**User:** Hi, I am trying to plan my trip and could use some help with a particular restaurant.
system response	SC-LSTM: Do you have a preference for area or price range?
	CSC-LSTM: Do you have a certain area or price range in mind?
	SC-GPT: Sure, what kind of food are you looking for and what area and price range?
	CSC-GPT: I would love to assist you! What type of food are you looking for and what price range?
context	**User:** Can you also help me find a train to Bishops Stortford?
	System: There are more than 70 trains to Bishops Stratford. Could you tell me what day you would like to leave, where you would like to leave from, and the times?
	User: I am leaving Cambridge on Sunday, can you please tell me the travel time on that ride?
	System: Sure, it is a 38 minute trip. Would you like a ticket?
	User: I need to leave after 14:45 also.
system response	SC-LSTM: The tr3600 leaves Cambridge at 15:29.
	CSC-LSTM: Tr3600 leaves Cambridge at 15:29. Would that works for you?
	SC-GPT: Tr3600 leaves Cambridge at 15:29.
	CSC-GPT: How about tr3600? It leaves Cambridge at 15:29.

Table 6: System response of different models given preceding dialogue context show that the contextual models: *CSC-LSTM* and *CSC-GPT*, can generate more natural responses that fit better to dialogue history.

Data Augmentation for Mental Health Classification on Social Media

Gunjan Ansari
JSS Academy of Technical Education, Noida, India
gunjanansari@jssaten.ac.in

Muskan Garg
Thapar Institute of Engineering & Technology
Patiala, Punjab, India
muskanphd@gmail.com

Chandni Saxena
The Chinese University of Hong Kong, Shatin, NT, Hong Kong
csaxena@cse.cuhk.edu.hk

Abstract

The mental disorder of online users is determined using social media posts. The major challenge in this domain is to avail the ethical clearance for using the user-generated text on social media platforms. Academic researchers identified the problem of insufficient and unlabeled data for mental health classification. To handle this issue, we have studied the effect of data augmentation techniques on domain-specific user-generated text for mental health classification. Among the existing well-established data augmentation techniques, we have identified Easy Data Augmentation (EDA), conditional BERT, and Back-Translation (BT) as the potential techniques for generating additional text to improve the performance of classifiers. Further, three different classifiers- Random Forest (RF), Support Vector Machine (SVM) and Logistic Regression (LR) are employed for analyzing the impact of data augmentation on two publicly available social media datasets. The experimental results show significant improvements in classifiers' performance when trained on the augmented data.

1 Introduction

Recent studies over mental health classification (Salari et al., 2020; Garg, 2021; Biester et al., 2021) convey that amid COVID-19 pandemic, the number of stress, anxiety and depression related mental disorders have increased. As per the recent survey, the rate of increase of mental disorders is more than those of physical health impacts on the Chinese population (Huang and Zhao, 2020). In this context, the early detection of psychological disorders is very important for good governance. It is observed that more than 80% of the people who commit suicide, disclose their intention to do so on social media (Sawhney et al., 2021). Clinical depression is the result of frequent tensions and stress. Further, prevailing clinical depression for a longer time period results in suicidal tendencies.

The information mining from social media helps in identifying stressful and casual conversations (Thelwall, 2017; Turcan and McKeown, 2019; Turcan et al., 2021). Many Machine Learning (ML) algorithms are developed in literature using both automatic and handcrafted features for classifying Microblog. The problem of data sparsity is under-explored for mental health studies on social media due to the sensitivity of data (Wongkoblap et al., 2017). Multiple ethical clearances are required for new developments in mental health classification. To deal with this issue of data sparsity, we have used data augmentation techniques to multiply the training data (Turcan and McKeown, 2019; Haque et al., 2021). The increase in training data may help to improve the hyper-parameter learning of textual features and thereby, reducing overfitting. Data Augmentation is the method of increasing the data diversity without collecting more data (Feng et al., 2021). The idea behind the use of Data Augmentation (DA) techniques is to understand the improvements in training classifiers for mental health detection on social media.

In this manuscript, the mental health classification is performed for two datasets to test the scalability of data augmentation approaches for mental healthcare domain. The classification of casual and stressful conversations (Turcan and McKeown, 2019), and classifying depression and suicidal posts (Haque et al., 2021) on social media. We select a rule based approach which preserves the original label and diversifies the text. To the best of our knowledge, this is the first attempt of stuffing additional data for mental health classification and there is no such study in the existing literature. The key contributions of this work are as follows:

- To determine the feasibility and the importance of data augmentation in the domain-specific study of mental health classification to solve the problem of data sparsity.

- The empirical study for different classification algorithms show significantly improved F-measure.

Ethical Clearance: We use limited, sparse and publicly available dataset for this study and so, no ethical approval is required from the Institutional Review Board (IRB) or elsewhere.

We organize rest of the manuscript in different sections. Section 2 describes the historical perspective of data augmentation and mental health classification on social media. We discuss the data augmentation methods and the architecture for experimental setups in Section 3. Section 4 elucidates the experimental results and evaluation over the proposed architecture of experimental setup which shows the significance and feasibility of data augmentation over mental health classification problems. Finally, Section 5 gives the conclusion and future scope of this work.

2 Related Work

Mental health classification can be quite challenging without the availability of sufficient data. Although the users' posts can be extracted from the social media platforms such as Reddit, Twitter and Facebook, annotating these posts is quite expensive. To address this issue, researchers have proposed different data augmentation techniques suitable for Natural Language Processing (NLP) which varies from simple rule-based methods to more complex generative approaches (Feng et al., 2021). The data augmentation tasks is categorized into conditional and unconditional augmentation task (Shorten et al., 2021).

2.1 Evolution of textual Data Augmentation

The unconditional data augmentation models like *Generative adversarial networks* (Goodfellow et al., 2014) and *Variational autoencoders* (Kingma and Welling, 2014) generates the random texts irrespective of the context. We do not use unconditional data augmentation for this task as it is required to preserve the context of the information as per the label. The conditional masking of a few tokens in the original sentence was observed to boost the classification performance in NLP tasks (Li et al., 2020; Wu et al., 2021). Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019), the pre-trained language models, are proposed with the objective to capture the left and right context in the sentence to generate the masked tokens. The pre-trained autoencoder model conditional BERT (Wu et al., 2019; Kumar et al., 2021) is used as a well-established technique for generating label compatible augmented data from the original data.

One of the simplest rule-based data augmentation techniques is proposed as Easy Data Augmentation (EDA) (Wei and Zou, 2019). The authors proposed four random operations such as *random insertion, random deletion, random swapping and random replacement* on the given text for generating new sentences. The experimental results give better performance on five benchmark text classification tasks (Wei and Zou, 2019), as the true labels of the generated text were conserved during the process of *data augmentation*. A graph based data augmentation is proposed for sentences using balance theory and transitivity to infer the pairs generated by augmentation of sentences (Chen et al., 2020). The sentence-based data augmentation is not suitable for the problem of mental health classification on Reddit data as the posts contain large paragraphs.

Back Translation (BT) or Round-trip translation is another augmentation technique which is used as a pipeline for text generation (Sennrich et al., 2015). The BT approach converts the A language of text to B language of text and then back to A language of the same text. This back-translation (Corbeil and Ghadivel, 2020) of data helps in diversifying the data by preserving its contextual information. Although, the interpolation techniques are proposed for data augmentation (Zhang et al., 2017), it is minimally used for textual data in existing literature (Guo et al., 2020).

In our work, we have studied the effect of all three different augmentation techniques- EDA, Conditional BERT and Back-translation to increase the size of training data for the task of mental health classification.

2.2 Mental Health Classification: Historical Perspective

The existing literature on mental health detection and analysis of social media data (Garg, 2021) shows the problem of automatic labeling as *noisy* *labels*. To handle this, either the label correction of noisy labels is required as shown in SDCNL (Haque et al., 2021) for manual labeling, or data augmentation (Chen et al., 2021). Since many existing datasets for mental health detection like RSDD, SMHD (Harrigian et al., 2020), CLPsych (Preoţiuc-Pietro et al., 2015) needs ethical clearance and are available only on request, we intend to pick small dataset with limited set of instances which are available in the public domain.

The Dreaddit dataset is manually labelled as stressful and casual conversation (Turcan and McKeown, 2019). In SDCNL dataset (Haque et al., 2021), the posts related to clinical depression and suicidal tendencies use similar words. Thus, we hypothesize that experimental results with data augmentation for classifying depression and suicidal risk may not generate well diversified data. In this manuscript, we use three data augmentation methods to text and validate the performance of the classifiers over both Dreaddit and SDCNL dataset.

3 Background: Data Augmentation Methods

Data augmentation (Feng et al., 2021) is a recent technique used for NLP to handle the problem of data sparsity by increasing the size of the training data without explicitly collecting the data. In this Section, we describe three potential textual data augmentation techniques, problem formulation, and architecture of the experimental setup.

3.1 Textual Data Augmentation

Out of many data augmentation tasks for NLP classification, very few are related to this problem domain of mental healthcare. This limitation is due to the presence of ill-formed (user-generated) text and the need to preserve the contextual information as per the label of the instances. To handle this issue, we use three different approaches. The first approach is based on NLP-based Augmentation technique (Wei and Zou, 2019), the second is based on conditional pre-trained language models such as BERT (Kumar et al., 2021) and the third approach is based on back translation (Ng et al., 2019). We briefly explain these methods in this section.

3.1.1 Easy Data Augmentation

In the previous work (Wei and Zou, 2019), NLP-based operations have been shown to achieve good results on text classification tasks. This method of data augmentation helps in diversifying the training samples while maintaining the class label associated with the post of a user at sentence level. The following four operations have been used in this work for augmenting the data:

- Synonym Replacement. Randomly n-words are chosen other than stop words from each sentence and replaced by one of its synonyms.
- Random Insertion. In this operation, a random synonym of a random word is inserted into a random position of a sentence for n number of times.
- Random Swap. Two words are randomly chosen in a sentence and swapped.
- Random Deletion. A word is deleted from a sentence with probability p.

3.1.2 Pre-Trained Language Models

Recently, deep bi-directional models have been used for generating textual data (Kobayashi, 2018; Song et al., 2019; Dong et al., 2017). These models are pre-trained with unlabelled text which can be fine tuned in autoencoder (Devlin et al., 2019), auto-regressive (Radford et al., 2019), or seq2seq (Lewis et al., 2019) settings. In *autoencoder settings*, a few tokens are randomly masked and the model is trained to predict alternative tokens. In *auto-regressive settings*, the model predicts the succeeding word according to the context. In *seq2seq settings*, the model is fine tuned on denoising autoencoder tasks. These transformers use associated class labels to generate the augmented text which helps in preserving its label. In this work, we adopt a framework[1] defined by (Kumar et al., 2021) and fine tune pre-trained BERT in auto-regressive settings.

3.1.3 Back Translation

Back translation (BT) is the data augmentation technique used for diversifying the information by changing the language of textual data to some language A and changing it back to its original language. In this experimental framework, we have used German as an intermediate language A. We use BT for the Microblogs by first converting it into German language using Neural Machine Translation (Ng et al., 2019) and then converting it back to

[1]https://github.com/amazon-research/transformers-data-augmentation

the English language. It is interesting to note that ill-formed and user-generated information is converted to the standard English language using BT and thus, spelling mistakes are reduced. Although the content is changed, contextual information is preserved.

3.2 Problem Formulation

Given a dataset D consisting of n-training samples where each sample is a text sequence x consisting of m-words and each sequence is associated with a label y. The objective is to generate an augmented data D_{syn} of n-synthetic samples using EDA, BERT and Back Translation.

3.2.1 AugEDA: Data augmentation using Easy Data Augmentation

In our work, 30% words of i^{th} training sample are randomly chosen for applying any one of the four EDA operation-Synonym Replacement, Random Insertion, Random Swap and Random Deletion (Wei and Zou, 2019). In synonym replacement, the chosen word is substituted by any one of the randomly selected synonym of this word from WordNet(Miller, 1995). In random insertion, j random positions are chosen for inserting random synonym of randomly chosen word out of m-words. In random swap, two words are randomly chosen from m-words and swapped with each other. A word is deleted with 10% probability in random deletion operation. The new sentence generated after applying any one of the lexical substitution method is added to the synthetic dataset D_{syn}. The process is repeated for n-training samples to create an augmented dataset of size n.

3.2.2 AugBERT: Data augmentation using BERT

We use the conditional BERT language model to generate the augmented data. We consider the label y and sequence $S = S_1, S_2...S_N$ of n-tokens to calculate the probability $p(t_i) = (.|y, S)$ of masked token t_i unlike masked language models that use only sequence S for predicting the probability of masked tokens. As defined by (Kumar et al., 2021), the conditional BERT model prepends associated label y to each sequence S in dataset D without adding it to the vocabulary of the model. For fine tuning of the model, some tokens of the sequence are randomly masked and the objective is to predict the original token according to the context of the sequence.

3.2.3 AugBT: Data augmentation using Back-Translation

To generate new textual data using Back-Translation, each of i^{th} training sample x_i is converted into a sentence y_i written in German language and then y_i is converted back to a sentence z_i in English. The generated sentence z_i is added to the augmented dataset D_{syn}. This process is repeated for n training samples to create an augmented dataset of n samples.

3.3 Architecture: Experimental Setup

The architecture of the experimental setup for augmenting domain-specific data of mental health classification from social media posts is shown in Figure 1. The Microblogs are given as an input for classifying the mental health of the users. The idea behind this approach is to generate some sequence of sentences and augment some more data for better training of classifiers. Thus, the number of instances are increased by using different data augmentation techniques.

The results are implemented for two publicly available mental health datasets, namely, Dreaddit and SDCNL. The dataset is divided into training and testing data. The training data is given as an input to the data augmentation methodologies, namely, EDA (Wei and Zou, 2019), Autoencoder conditional BERT (Wu et al., 2019) and Back-Translation (Ng et al., 2019). These three approaches are well established approaches for data augmentation in classification of the textual data. The original training data is almost doubled in the process of the data augmentation. The original and augmented data are fed to different machine learning classifiers for results and analysis.

4 Experimental Results and Evaluation

In this section, we discuss the datasets and the experimental results. We further analyze results for data diversity and statistical significance of the classifiers over augmented data as compared to the original data.

4.1 Dataset

The idea behind this study is to improve the training parameters of the classifier by removing the limitation of data sparsity. The two sparse datasets which are used for domain-specific data augmentation are Dreaddit [2] (Turcan and McKeown, 2019)

[2]http: //www.cs.columbia.edu/˜eturcan/data/ dreaddit.zip.

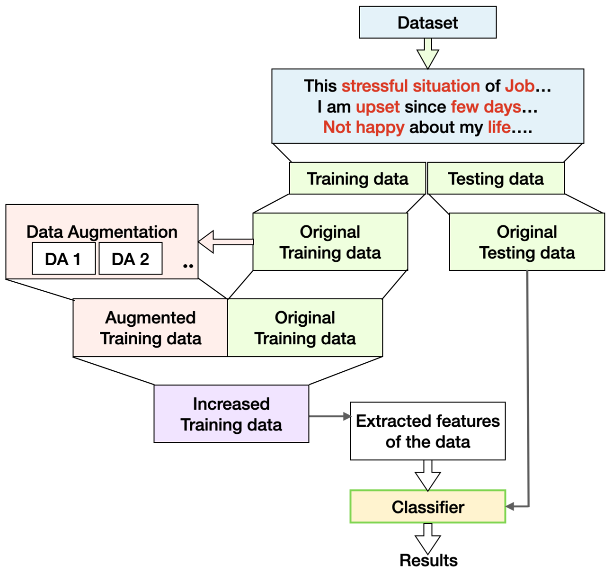


Figure 1: The Architecture of Experimental Setup for Data Augmentation

and SDCNL[3] (Haque et al., 2021) from existing literature are explained in this Section.

4.1.1 Dreaddit dataset

The Dreaddit dataset(Turcan and McKeown, 2019) consists of lengthy posts in five different categories and is used for classifying stressful posts from casual conversations. The categories of subreddits selected by authors having stressful conversations are interpersonal conflicts, mental illness (anxiety and PTSD), financial and social.

Dataset	Stress	Non-Stress
Training data	1488	1350
Testing data	369	346

Table 1: Dreaddit Dataset Statistics

Out of total 187444 posts scraped from these five categories, the authors have manually labelled 3553 Reddit posts. While selecting the posts for annotation, the authors selected those segments whose average token length was greater than 100. The average tokens per post in this dataset is 420 tokens. This statistics of the Dreaddit dataset is shown in Table 1.

4.1.2 SDCNL dataset

The SDCNL dataset(Haque et al., 2021) is scrapped from Reddit social media platform from two subreddits: *r/SuicideWatch* and *r/Depression* to carry out the study for classifying posts into depression specific or suicide specific. This dataset contains 1895 posts containing 1517 training samples and 379 testing samples. The dataset contains title, selftext and megatext of the reddit tweets along with other fields.

Dataset	Depression	Suicide
Training data	729	788
Testing data	186	193

Table 2: SDCNL Dataset Statistics

In this dataset, 729 out of 1517 instances are labelled as depression specific posts as shown in Table 2. The dataset is manually labelled to reduce noisy automated labels. The idea behind using this data is that we hypothesise that this dataset is even more complex than the Dreaddit dataset due to the presence of similar domain-specific words in posts.

4.2 Experimental Setup

The original and the augmented dataset used for experimentation is quite noisy as the posts used in this data is user-generated natural language text expressing the feelings of the writer. The preprocessing steps are applied using the NLTK library[4] of Python (Bird, 2006). The data is transformed before applying the supervised learning models employed in this work. The posts are long paragraphs, so in the first step the data is tokenized into sentences and then sentences are further tokenized into words. After removal of stopwords, punctuations,unknown characters from the extracted tokens, we use stemming and lemmatization to extract the root words.

After pre-processing of the data, it is transformed to a feature vector using Term Frequency- Inverse Document Frequency (TF-IDF), Word2Vec (Goldberg and Levy, 2014) and Doc2Vec (Lau and Baldwin, 2016). Word2Vec embedding and Doc2Vec embedding provides dense vector representation of data while capturing its context. In this research work, the Gensim library[5] is used to learn word embeddings from the training corpus using skip-gram algorithm. A vector of 300 dimensions is chosen and default settings of Word2Vec and Doc2Vec models are used for experiments and evaluation.

The learning based classifiers which are used for this research work are the Logistic Regression

[3]https://github.com/ayaanzhaque/SDCNL

[4]https://www.nltk.org/

[5]https://pypi.org/project/gensim/

(LR), the Support Vector Machine(SVM), and the Random Forest (RF) with the default settings of scikit-learn[6] (sklearn) library of Python. The hardware configuration of the system which is used to perform this study is 2.6 GHz 6-core Intel Core i7, Turbo Boost up to 4.5 GHz, with 12 MB shared L3 cache.

4.3 Experimental Results

We reference (Kumar et al., 2021) for implementation [7] and use AugBERT, AugEDA, and AugBT on two datasets- Dreaddit and SDCNL. The dataset is divided into 75% training and 25% testing set and the value of Precision (P), Recall(R) and F1 score (F1) are computed on the testing samples to evaluate the performance of the classifiers with and without domain -specific data augmentation for mental health classification. Table 3 and Table 4 presents the results achieved for original and augmented data for Dreaddit and SDCNL using three different classifiers, namely, Logistic regression (LR), Support Vector Machine (SVM) and Random Forest (RF), respectively.

4.3.1 Experimental Results for Dreaddit

As observed from Table 3, the F1 score showed an average improvement of around 1.4% achieved by all models with AugBERT as compared to the original training dataset. It is also found that the AugEDA gives maximum improvement of around 4% when Word2Vec and Doc2Vec embeddings were employed with LR. Also, there is negligible improvement in the results with AugBT.

4.3.2 Experimental Results for SDCNL

In this Section, the results of the experimental study are presented for the SDCNL dataset. As observed from Table 4, the average improvement of around 2.3% is observed for all the models as per F1 score with AugBERT. The AugEDA shows maximum improvement of more than 5% when Word2Vec and Doc2Vec embeddings were employed with RF. The results also indicate a minor improvement of around $1-2\%$ when classifiers employed Doc2Vec and TF-IDF embeddings for representing augmented data using Back Translation.

Due to increase in the size of augmented data, the input vector representations using TF-IDF requires higher computational time as compared to other embeddings. Thus, a few results are shown empty in Table 3 and Table 4. In healthcare, more *precise* results are expected than *recall* which means that the content which is identified as stressful must be correct and matters more than diagnosing the total number of correct instances. Thus, precision must improve more than recall values. We have considered these nuances to examine the results of classifiers and found that Logistic Regression gives improved results with the Doc2Vec encoding scheme.

4.4 Data Diversity of Augmented Data

The diversity of the generated data by different augmentation techniques are measured by the Bilingual Evaluation Understudy (BLEU) score (Papineni et al., 2002). The BLUE score ranges between 0 and 1. The lower the value, the better is the diversity in the data. Thus, the BLEU score is computed by comparing n-grams of both original and generated text where $n = 2$.

As observed from Table 5, the BLEU score for augmented data varies from 82% - 99%. The training samples are multiplied by 1.75 to 2.0 times for data augmentation approaches. The data for AugBERT is more diversified and thus, the results are significantly improved for AugBERT rather than AugEDA and AugBT as evident from Table 3 and Table 4. The experimental results show that the samples are upto 18% more diverse than those of original training samples for AugBERT over the Dreaddit dataset. However, the least data diversity is observed for AugEDA and AugBT over the SDCNL dataset.

4.5 Statistical Significance

In this Section, to understand the importance of generating more instances in training data is performed using three different data augmentation techniques. The statistical student's t-test was used to test the significance of the improvement in classifier using augmented data with $p-value$ as 0.05, 0.10, and 0.15. The resulting value for t-test in Dreaddit and SDCNL over AugBERT is obtained as 0.00033 and 0.09241 which shows the overall significant improvements with 5% and 10% significant levels, respectively. The results are improved in 83%, and 66% in the cases of different encoding vectors and classifiers which are used as learning based algorithms for AugBERT and AugEDA data augmentation techniques, respectively.

[6]https://scikit-learn.org/stable/
[7]https://github.com/varunkumar-dev/TransformersDataAugmentation

Methods used	**Original**			**AugBERT**			**AugEDA**			**AugBT**		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
RF+Word2Vec+TFIDF	0.68	0.84	0.75	0.69	0.84	**0.76**	0.68	0.81	0.74	0.67	0.84	0.74
SVM+Word2Vec+TFIDF	0.69	0.75	0.72	0.69	0.79	**0.74**$^+$	0.69	0.79	**0.74**$^+$	0.74	0.66	0.69
LR+Word2Vec+TFIDF	0.71	0.78	0.74	0.71	0.79	**0.75**	0.72	0.79	**0.75**	0.71	0.77	0.74
RF+Doc2Vec	0.65	0.78	0.71	0.62	0.83	0.71	0.65	0.80	**0.72**	0.63	0.82	**0.72**
SVM+Doc2Vec	0.74	0.76	0.75	0.73	0.78	**0.76**	0.74	0.73	0.73	0.73	0.75	0.74
LR+Doc2Vec	0.73	0.75	0.74	0.73	0.78	**0.76**$^+$	0.72	0.73	0.72	0.72	0.76	0.74
RF+Word2Vec+Doc2Vec	0.85	0.67	0.75	0.67	0.86	0.75	0.68	0.86	**0.76**	0.66	0.84	0.74
SVM+Word2Vec+Doc2Vvec	0.75	0.68	0.71	0.70	0.74	**0.72**	0.73	0.71	**0.72**	0.71	0.74	**0.72**
LR+Word2Vec+Doc2Vec	0.76	0.69	0.72	0.71	0.76	**0.73**	0.75	0.78	**0.76**$^+$	0.71	0.76	**0.73**
RF+TFIDF	0.70	0.73	0.71	0.69	0.79	**0.74**	0.67	0.84	**0.74**$^+$	-	-	-
SVM+TFIDF	0.79	0.68	0.73	0.74	0.78	**0.76**$^+$	0.70	0.73	0.72	-	-	-
LR+TFIDF	0.68	0.82	0.74	0.70	0.80	**0.75**	0.76	0.75	**0.75**	-	-	-

Table 3: Classification Results on Dreaddit Dataset: Precision(P), Recall(R), F-measure(F1) score on the Original and Augmented Datasets using BERT, EDA and BackTranslation. Text in bold shows the maximum F1 score achieved by the model. '-' indicates no results. '+' indicates significantly different results using statistical t-test.

Methods used	**Original**			**AugBERT**			**AugEDA**			**AugBT**		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
RF+Word2Vec+TFIDF	0.68	0.67	0.67	0.69	0.69	**0.69**$^+$	0.63	0.66	0.65	0.65	0.68	0.67
SVM+Word2Vec+TFIDF	0.69	0.67	0.68	0.65	0.66	0.66	0.67	0.70	0.69	0.63	0.67	0.65
LR+Word2Vec+TFIDF	0.63	0.70	0.66	0.67	0.73	**0.70**$^+$	0.65	0.69	**0.67**	0.61	0.67	0.64
RF+Doc2Vec	0.65	0.57	0.61	0.63	0.51	0.56	0.64	0.55	0.60	0.59	0.57	0.58
SVM+Doc2Vec	0.65	0.66	0.65	0.66	0.73	**0.70**$^+$	0.66	0.70	**0.68**$^+$	0.65	0.69	**0.67**$^+$
LR+Doc2Vec	0.65	0.67	0.66	0.68	0.76	**0.71**$^+$	0.68	0.71	**0.69**$^+$	0.66	0.69	**0.68**$^+$
RF+Word2Vec+Doc2Vec	0.63	0.64	0.63	0.63	0.58	0.60	0.67	0.69	**0.68**$^+$	0.66	0.67	**0.67**$^+$
SVM+Word2Vec+Doc2Vec	0.65	0.67	0.66	0.64	0.70	**0.67**	0.60	0.66	0.66	0.62	0.64	0.63
LR+Word2Vec+Doc2Vec	0.64	0.64	0.64	0.64	0.73	**0.68**$^+$	0.59	0.65	0.62	0.61	0.65	0.63
RF+TFIDF	0.61	0.85	0.71	0.63	0.81	0.71	-	-	-	-	-	-
SVM+TFIDF	0.71	0.75	0.73	0.67	0.85	**0.75**$^+$	0.76	0.73	**0.75**$^+$	0.71	0.77	**0.74**
LR+TFIDF	0.70	0.77	0.73	0.68	0.85	**0.76**$^+$	0.76	0.75	**0.75**$^+$	0.71	0.77	**0.74**

Table 4: Classification Results on SDCNL Dataset: Precision(P), Recall(R), F-measure(F1) score on Original and Augmented Datasets using BERT, EDA and Back Translation. Text in bold shows the maximum F1 score achieved by the model. '-' indicates no results.'+' indicates significantly different results using statistical t-test.

	Dreaddit	**SDCNL**
AugEDA	0.97	0.99
AugBERT	0.82	0.97
AugBT	0.88	0.99

Table 5: Data Diversity using BLEU Score

Dreaddit	**AugBERT**	**AugEDA**	**AugBT**
t-test	-4.69041	1.07605	0.75593
p-value	0.00033	0.15247	0.23568

Table 6: Statistical Significance of overall results with Original Data

4.5.1 Statistical Significance for Dreaddit

It is evident from Table 6 that AugBERT and AugEDA show significantly improved results and there is no effect of AugBT over domain-specific data augmentation for mental health.

On drilling down the results, it is observed that the AugBERT based augmented results for SVM classifier are significantly better than the other classification techniques. Some more significant improvements with the use of LR classifier is observed as shown in Table 3 with as high as 5% for AugEDA. The variation of improvement in results ranges upto 4.1%, 5.5% and 1.3% for AugBERT, AugEDA and AugBT, respectively.

4.5.2 Statistical Significance for SDCNL

The significant improvements over SDCNL dataset is observed on the basis of $p - value$ as 0.05, 0.10 and 0.15 as shown in Table 7. The results have shown that the AugBERT and AugEDA gives better results for 10% variation in results and validates the hypothesis that the augmented data gives significant improvements over the original dataset.

SDCNL	AugBERT	AugEDA	AugBT
t-test	-1.42426	-1.6361	0.25118
p-value	0.09241	0.06644	0.40338

Table 7: Statistical Significance of overall results with Original Data

Similar to the Dreaddit observations, the significant improvements with LR classifier are observed for classifying mental health into clinical depression and suicidal tendencies. On the contrary, SVM with Doc2Vec shows much better results with AugBERT, AugEDA and AugBT.

5 Conclusion

In this work, we use the data augmentation approach for mental health classification on two different social media datasets. The experimental results using Logistic Regression classifier and Doc2Vec embedding shows significant improvements in F1 score and Precision with AugBERT. To tackle the problem of data sparsity and support the automation of the 3-Step theory over social media data (Klonsky and May, 2015), the data augmentation over mental healthcare may give remarkable results. In future, we are planning to use other domain-specific libraries and neural machine translation for explainable and conditional data augmentation.

References

Laura Biester, Katie Matton, Janarthanan Rajendran, Emily Mower Provost, and Rada Mihalcea. 2021. Understanding the impact of covid-19 on online mental health forums. *ACM Transactions on Management Information Systems (TMIS)*, 12(4):1–28.

Steven Bird. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 69–72.

Hannah Chen, Yangfeng Ji, and David Evans. 2020. Finding friends and flipping frenemies: Automatic paraphrase dataset augmentation using graph theory. *arXiv preprint arXiv:2011.01856*.

Jiaao Chen, Derek Tam, Colin Raffel, Mohit Bansal, and Diyi Yang. 2021. An empirical survey of data augmentation for limited data learning in nlp. *arXiv preprint arXiv:2106.07499*.

Jean-Philippe Corbeil and Hadi Abdi Ghadivel. 2020. Bet: A backtranslation approach for easy data augmentation in transformer-based paraphrase identification context. *arXiv preprint arXiv:2009.12452*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. *arXiv preprint arXiv:1708.06022*.

Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for nlp.

Muskan Garg. 2021. Quantifying the suicidal tendency on social media: A survey. *arXiv preprint arXiv:2110.03663*.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks.

Demi Guo, Yoon Kim, and Alexander M Rush. 2020. Sequence-level mixed sample data augmentation. *arXiv preprint arXiv:2011.09039*.

Ayaan Haque, Viraaj Reddi, and Tyler Giallanza. 2021. Deep learning for suicide and depression identification with unsupervised label correction. *arXiv preprint arXiv:2102.09427*.

Keith Harrigian, Carlos Aguirre, and Mark Dredze. 2020. Do models of mental health based on social media data generalize? In *Proceedings of the 2020 conference on empirical methods in natural language processing: findings*, pages 3774–3788.

Yeen Huang and Ning Zhao. 2020. Generalized anxiety disorder, depressive symptoms and sleep quality during covid-19 outbreak in china: a web-based cross-sectional survey. *Psychiatry research*, 288:112954.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes.

E David Klonsky and Alexis M May. 2015. The three-step theory (3st): A new theory of suicide rooted in the "ideation-to-action" framework. *International Journal of Cognitive Therapy*, 8(2):114–129.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2021. Data augmentation using pre-trained transformer models.

Jey Han Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Kun Li, Chengbo Chen, Xiaojun Quan, Qing Ling, and Yan Song. 2020. Conditional augmentation for aspect term extraction via masked sequence-to-sequence generation. *arXiv preprint arXiv:2004.14769*.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook fair's wmt19 news translation task submission. *arXiv preprint arXiv:1907.06616*.

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.

Daniel Preoţiuc-Pietro, Maarten Sap, H Andrew Schwartz, and Lyle Ungar. 2015. Mental illness detection at the world well-being project for the clpsych 2015 shared task. In *Proceedings of the 2nd Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*, pages 40–45.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Nader Salari, Amin Hosseinian-Far, Rostam Jalali, Aliakbar Vaisi-Raygani, Shna Rasoulpoor, Masoud Mohammadi, Shabnam Rasoulpoor, and Behnam Khaledi-Paveh. 2020. Prevalence of stress, anxiety, depression among the general population during the covid-19 pandemic: a systematic review and meta-analysis. *Globalization and health*, 16(1):1–11.

Ramit Sawhney, Harshit Joshi, Lucie Flek, and Rajiv Shah. 2021. Phase: Learning emotional phase-aware representations for suicide ideation detection on social media. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2415–2428.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

Connor Shorten, Taghi M Khoshgoftaar, and Borko Furht. 2021. Text data augmentation for deep learning. *Journal of big Data*, 8(1):1–34.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*.

Mike Thelwall. 2017. Tensistrength: Stress and relaxation magnitude detection for social media texts. *Information Processing & Management*, 53(1):106–121.

Elsbeth Turcan and Kathleen McKeown. 2019. Dreaddit: A reddit dataset for stress analysis in social media. In *LOUHI@EMNLP*.

Elsbeth Turcan, Smaranda Muresan, and Kathleen McKeown. 2021. Emotion-infused models for explainable psychological stress detection. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2895–2909.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*.

Akkapon Wongkoblap, Miguel A Vadillo, and Vasa Curcin. 2017. Researching mental health disorders in the era of social media: systematic review. *Journal of medical Internet research*, 19(6):e228.

Xing Wu, Shangwen Lv, Liangjun Zang, Jizhong Han, and Songlin Hu. 2019. Conditional bert contextual augmentation. In *International Conference on Computational Science*, pages 84–95. Springer.

Yuan Wu, Diana Inkpen, and Ahmed El-Roby. 2021. Conditional adversarial networks for multi-domain text classification. *arXiv preprint arXiv:2102.10176*.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

A Appendix

Samples of original and augmented data

Original data	Augmented data (AugEDA)	Augmented data(AugBERT)	Augmented data(AugBT)
He said he could run some more tests, but he didn't think it would help.	he talk about my said run tests but he didnt think would help	he said he tried run some more medicine, but he weren't think it would help.	he said he could run some clinical tests, but he didn't think it would okay.
Is always adamant about keeping contact with the people she cheated with.	is always headstrong about contact with the people me cheated with	then rather adamant about keeping contact featuring the people she cheated with	is always adamant by keeping track featuring strange people she cheated with
It seemed like a circulation problem, and I panicked and of course ended up in the ER again.	it seemed as a circulation problem i panicked and of course finish up in er again	many said like a relationship collapsed, and i, and same course ended up entering the er again	it seemed had mostly circulation problem, and i panicked and of course ended up in er again.

VAE based Text Style Transfer with Pivot Words Enhancement Learning

Haoran Xu [1,2], Sixing Lu [2], Zhongkai Sun [2], Chengyuan Ma [2], Chenlei Guo [2]
[1]Johns Hopkins University, [2]Amazon Alexa AI

hxu64@jhu.edu
{cynthilu, zhongkas, mchengyu, guochenl}@amazon.com

Abstract

Text Style Transfer (TST) aims to alter the underlying style of the source text to another specific style while keeping the same content. Due to the scarcity of high-quality parallel training data, unsupervised learning has become a trending direction for TST tasks. In this paper, we propose a novel VAE based Text Style Transfer with pivOt Words Enhancement leaRning (VT-STOWER) method which utilizes Variational AutoEncoder (VAE) and external style embeddings to learn semantics and style distribution jointly. Additionally, we introduce pivot words learning, which is applied to learn decisive words for a specific style and thereby further improve the overall performance of the style transfer. The proposed VT-STOWER can be scaled to different TST scenarios given very limited and non-parallel training data with a novel and flexible style strength control mechanism. Experiments demonstrate that the VT-STOWER outperforms the state-of-the-art on sentiment, formality, and code-switching TST tasks [1].

1 Introduction

Text style transfer (TST) is an important task in the natural language generation area, aiming to control the certain manner of the semantics style expressed in the generated text. Such styles include but not limit to emotion, humor, politeness, formality, and code-switching. For instance, sentiment transfer is widely seen in sentiment analysis for reviewing comments (e.g., yelp, twitter), and targets on converting the original negative/positive comment into a new comment with same topic but opposite sentiment (Hu et al., 2017; Shen et al., 2017); formality transfer is commonly used in documenting, aims at transferring the informal oral expression into a formal written expression (Jin et al., 2020). In this paper, we also consider code-switching as a style transfer task, which has not been explored by previous works. Code-switching is a complicated linguistic phenomenon where a speaker alternates between two or more languages in one utterance, either inter-sentential or intra-sentential. The code-switching transfer is a more challenging task considering cross-lingual alignment and limited available training data in nature. Examples of these three style transfer tasks are shown in the Figure 1.

Because of the scarcity of high-quality parallel training data, unsupervised learning has become the mainstream for TST tasks. Existing works on unsupervised TST learning can be roughly categorized into *Disentanglement* (Shen et al., 2017; Hu et al., 2017; Fu et al., 2018; John et al., 2019) and *Style Attribute Rewriting* (Lample et al., 2019; Dai et al., 2019; Yi et al., 2020). Disentanglement approaches strip style features from the content and incorporate the content features with the target style representation. However, researchers become less focus on disentanglement methods after Locatello et al. (2019) theoretically proved disentanglement approaches are impossible to represent style fully with unsupervised learning. The style attribute rewriting enforces the model to focus on style-independent words by cycle reconstruction and rewriting the style attributes with style embeddings. Dai et al. (2019) firstly proposed style transfer model based on the transformer architecture along with target style information. Lample et al. (2019) reported that a good decoder can generate the text with the desired style by rewriting the original style. However, the style strength of the generated sentences cannot be easily adjusted in above mentioned works.

Variational autoencoder (VAE) is firstly proposed by (Kingma and Welling, 2014) for gener-

Work done during an internship at Amazon Alexa AI.

[1]The code is available at https://github.com/felixxu/VT-STOWER.

ation by formatting the latent distribution instead of feeding a single latent feature to the decoder. Many TST models have been benefited from the architecture of VAE. Bowman et al. (2016); John et al. (2019) showed that the latent space learned by VAE is considerably smoother and more continuous than the one learned by Deterministic Autoencoder (DAE). Hu et al. (2017) proposed a new neural generative model that combines VAE and holistic attribute discriminators for effective imposition of the style semantic structures. In this paper, we also leverage the VAE and propose a novel method called VAE based Text Style Transfer with pivOt Words Enhancement leaRn-ing (VT-STOWER) for TST tasks. VT-STOWER utilizes both VAE and style embeddings to jointly learn the distribution of content and style features. More importantly, we boost the performance of TST tasks much more by inventing pivot words enhancement learning. Compared with other style-transfer methods, our proposed VT-STOWER has a bunch of advantages. In general, the advantages and contributions of the VT-STOWER can be summarised as follows:

Negative	they were **dry** and truly **tasteless**.
Positive	they were **tasty** and truly **delicious**.
Informal	**u 'll** find a man who really deserves **u** one day.
Formal	**you will** find a man who really deserves **you** one day.
Hindi	कया आपको एनीमशन **फिलम** पसद ह ? (Do you like animation movies?)
Code-switch	कया आपको एनीमशन **movies** पसद ह ? (Do you like animation movies?)

Figure 1: Examples of different TST. Including sentiment style transfer (negative $\leftrightarrow$ positive), formality style transfer (informal $\leftrightarrow$ formal), code-switch style transfer (single language $\leftrightarrow$ code-switch sentence).

- VT-STOWER integrates the advantages of both VAE and style embeddings. The former catches continuous style expression distribution in language itself while the latter differentiates embedding between original style and target style.

- VT-STOWER has the flexibility to adjust the target style strength by granting different weights to the auxiliary target style embedding; This allows VT-STOWER to better migrate to different style transfer scenarios, which is rarely studied in previous style-transfer work.

- With the *pivot words masking enhancement* mechanism, VT-STOWER is able to focus more on the pivot words (certain words that can determine the style of the sentence) and be aware of which words have higher probability to be transferred in the TST. This enhancement significantly improves the transfer accuracy while maintaining original topic.

- VT-STOWER can be easily scaled to different types of TST tasks. To the best of our knowledge, we are the first to consider code-switching in perspective of style transfer and demonstrate that VT-STOWER can be successfully applied to the Hindi-Hinglish code-switching transfer. Therefore, we provide more potential solutions for the the code-switching problems beyond translation by which translating from single language to code-switching expression is very hard given limited training data.

- We evaluate VT-STOWER on the benchmark dataset of sentiment, formality transfer tasks, and the code-switching style transfer. Experimental results on all tasks demonstrate better overall performance against state-of-the-art methods, which highlights effectiveness and wide application of VT-STOWER.

2 Proposed Method

The training of VT-STOWER consists of two stages. The training stage I is a VAE reconstruction task in which the input text x will be reconstructed together with external style embeddings. The latent space of content distribution is learned by VAE, and the original and target style mapping will be learned and saved in style embeddings. The trained VAE and style embeddings will also be utilized in the second training stage.

To make the style transfer focus on pivot words (e.g., emotional words in sentiment TST) while maintaining other words unchanged (so that the fluency and semantics can be largely preserved), we fine-tune the VAE with pivot word masking in training stage II. The masking is based on the probability distribution of pivot words for specific styles, which is learned from a style classification task.

In the inference stage, VT-STOWER uses the learned external target style embeddings to adjust the sampled latent vector of the original input to the target style. The adjusted sentence vector will then be input to the decoder to generate the target style text.

2.1 Training Stage I: VAE & Style Embeddings

Figure 2a presents the details of training stage I. Given a sentence x whose style type is known, we firstly extract the contextualized vectors through a pre-trained language model as the input to the VAE model, since a pre-trained language model (such as `RoBerta` (Liu et al., 2019) and `XLM-R` (Conneau et al., 2020)) can improve the performance of the downstream models, especially when the training data size is small (Peters et al., 2018). After that, similar to typical VAE structure from Bowman et al. (2016); John et al. (2019), a multi-layer transformer is used as the encoder to encode x to a mean vector $u \in \mathbb{R}^d$ and a variance vector $\Sigma \in \mathbb{R}^d$ to construct a latent distribution $\mathcal{N}(\mu, \Sigma)$. d represents the dimension of the latent space. z is the vector sampled from the latent distribution and will be input to the decoder (which is also a multi-layer transformer) to reconstruct the original text. The latent distribution is assumed to be a normal distribution $\mathcal{N}(0, \mathrm{I})$. The standard loss function of the VAE model is defined as:

$$\mathcal{L}_{vae} = -\mathbb{E}_{q(z|x)}[\log p(x|z)] + \beta \cdot \mathbb{KL}(q(z|x) \parallel p(z)) \quad (1)$$

where the first term represents the likelihood of the reconstruction of the original text x while the second term is the Kullback–Leibler (KL) divergence between the latent distribution and standard normal distribution. $p(z)$ represents the prior which is the standard normal distribution $\mathcal{N}(0, \mathrm{I})$, and $q(z|x)$ is the posterior distribution in the form of $\mathcal{N}(\mu, \Sigma)$. β is the hyperparameter balancing the learning capacity between self-reconstruction and style features (Higgins et al., 2016).

Style embeddings are also learned in this training stage. Instead of disentangling style attributes from latent features (Shen et al., 2017; Hu et al., 2017; Fu et al., 2018; John et al., 2019), we utilize external style embeddings to learn the original and target style representations. The advantage of external style embeddings is that they can avoid separating latent feature which leads to the lower capacity of vector representation (Dai et al., 2019), and can differentiate the space of different styles. The set of style embeddings is defined as $S = \{s_1, s_2, \cdots, s_k\}, s_i \in \mathbb{R}^{k \times d}$, where k is the number of styles (k is commonly to be 2 in TST tasks). Style embeddings are generated by a linear forward network whose output dimension is d. This style embedding network is randomly initialized and will be updated by minimizing the similarity between the style embeddings and latent feature of the input instances.

To minimize such similarity, we calculate the cosine similarity between style embeddings s_i ($1 \leq i \leq k$) and sampled latent feature z as the style loss. The assumption is that the style embedding should be highly related to the latent feature encoded from the sentence which belongs to the same style, e.g., the distance between positive style embedding and latent vector encoded from positive sentence should be close to 1, while the distance should be 0 between positive style embedding and latent vector from the negative input sentence. The style loss is defined as follows:

$$\mathcal{L}_{style} = -\sum_{i=1}^{k} d_i \log(\sigma(\cos(s_i, \mathrm{sg}(z)))) \quad (2)$$

For brevity, we only present the loss for a single style sentence, where d_i represents the ground truth distance. Specifically, if i_{th} style is the style of the input sentence, $d_i = 1$, otherwise, $d_i = 0$. $\sigma(\cdot)$ here is the sigmoid activation function which controls the range of cosine similarity between 0 and 1. $\mathrm{sg}(\cdot)$ is the 'stop gradient' function, e.g., the feature $\mathrm{sg}(z)$ is extracted through the latent distribution and used as an independent constant vector for computing the $\mathcal{L}_{style}$. The VAE loss is slightly modified from Equation 1 by adding style embedding to hint decoder the style of sentences to be generated.

$$\mathcal{L}_{vae} = -\mathbb{E}_{q(z|x)}[\log p(x|z + \mathrm{sg}(s_x))] + \beta \cdot \mathbb{KL}(q(z|x) \parallel p(z)) \quad (3)$$

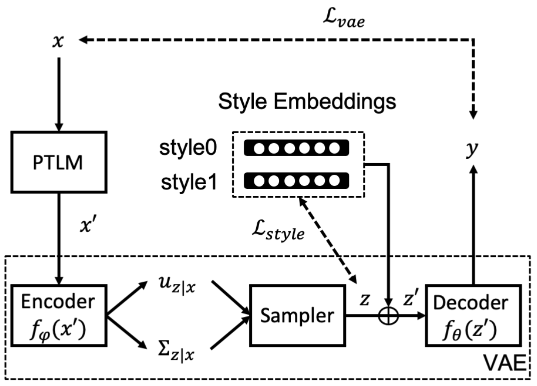
where s_x is the style embedding of sentence x. Similarly, the s_x is also used as a constant vector. Therefore, the total loss function is then defined as:

$$\mathcal{L}_{total} = \lambda_{vae}\mathcal{L}_{vae} + \lambda_{style}\mathcal{L}_{style} \quad (4)$$

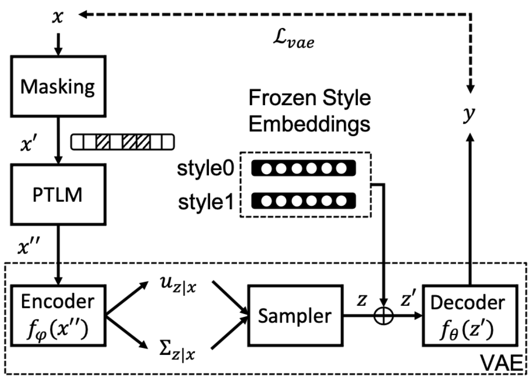
where λ_{vae} and λ_{style} are penalty weights, which are hyperparameters to balance between VAE loss and style embeddings loss.

2.2 Training Stage II: Pivot Words Masking

In Stage I, we co-train VAE & style embeddings where we show how to leverage learned style embeddings to further improve the VAE model. In stage II, We further enhance the model by masking pivot words to prompt decoder to focus on pivot



(a) Training stage I: VAE & Style Embeddings



(b) Training stage II: Pivot Words Enhancement

Figure 2: Workflow of two training stages. a) Training stage I: VAE & style embeddings training. The VAE structure learns to reconstruct the inputs sentence x, and the style embeddings learn the vector representation of each style. PTLM represents Pre-Trained Language Model. b) Training stage II: pivot words masking training. The VAE is further fine-tuned with similar reconstruction task with additional pivot words masking. The frozen style embeddings are added to the latent vector to reconstruct the original sentence. We frozen the style embeddings in this step since the style-related pivot words have higher possibility to be masked and latent vector loses the style information which is the key for style embeddings training.

words, because certain style-related words play crucial roles in TST (Fu et al., 2019). For instance, the pivot word of the sentence 'I am disappointed with the restaurant' in sentiment transfer is 'disappointed' because this word contributes the most to the negative sentiment. However, other words such as "I, was" are anchor words, which are unrelated to the sentiment but affect the semantics thus should be unchanged during the style transfer. Therefore, this stage of training is important to enhance the model ability in transferring pivot words while keeping anchor words. This stage cannot be merged into training stage I because 1) in this stage style embeddings have no visibility to the style-related pivot words so that the style information is hard to be learned; 2) the style embeddings learned in training stage I have auxiliary function in helping reconstructing masked pivot words during fine-tuning the VAE.

However, randomly masking words in input sentence and only relying on style embeddings to emphasize the pivot words does not achieve ideal results. A more efficient way is to learn which words are more possible to be pivot words for a specific style, and mask them based on the probability. Similar to Sudhakar et al. (2019), we utilizes the importance score distribution to indicate the possibility of words being pivot (a pivot word has a higher score). Such importance score distribution is achieved from the attention weights of a style classifier. Specifically, we train a style classifier based on a pre-trained language model, appending with a softmax layer over the attention stack of the first token. The first token is usually a special symbol that represents the beginning of the sentence (e.g., '<s>'), and also collects other tokens' attention weights that correspond to their significance in identifying the style of the input sentence. The importance score of a token w in the input sentence x is defined as follows:

$$\alpha(w) = \frac{1}{L}\sum_{i=1}^{L} \text{softmax}_{w\in x}(\frac{Q_{<s>,i}K_{w,i}^T}{\gamma}) \quad (5)$$

where L is the number of attention heads. Q, K are quires and keys in the final layer of the language model (Vaswani et al., 2017). Their subscript $< w, i >$ represents the vector of token w in i_{th} head. γ is a hyperparameter ranging in (0,1) to adjust the sharpness of the score distribution (smaller means sharper).

After we get the pivot words probability, we mask words in the input sentences based on this importance score distribution. Specifically, every token x_i is assigned a random number p_i, conforming to the uniform distribution $p_i \sim$ uniform[0,1]. Tokens are masked into a special symbol '<mask>' if their assigned number is smaller than the score ($p_i < \alpha(x_i)$) so that words that possess higher important scores have higher probability to be masked. Following the previous example, the input sentence would be masked as 'I was <mask> with the restaurant'. In this way, masked sentence preserves the content but with style attributes removed. Then the VAE model is fine-tuned to reconstruct masked sentence to the original sentence by adding the corresponding style embedding to the latent feature. The loss function is defined as follows:

$$\mathcal{L}_{vae} = -\mathbb{E}_{q(z|x)}[\log p(x|z + \text{sg}(s'_x))] + \beta \cdot \mathbb{KL}(q(z|x) \parallel p(z)) \quad (6)$$

where s'_x is the style embedding which has the same style as input x. Note that we do not update style embeddings in this stage because the style embeddings are used for assisting fine-tuning the decoder with style information, and their general style representation should not be impacted by the pivot words reconstruction loss. Moreover, to prevent the latent space of VAE from shifting or distorting to unreasonable distribution that only describes masked sentences, we conduct pivot words masking in randomly 50% of sentences.

Although Madaan et al. (2020) has a similar method tagging the source style phrases and generating the target style sentences by using n-gram tf–idfs, the core differences of our stage II method are: 1) each word has a probability of being masked calculated by the attention scores of the stacked classifier on a language model, which leads to a smooth word masking probability distribution; 2) VAE decoder reconstructs the masked sentences using both information of latent space and external style embeddings.

2.3 Inference stage

In the inference stage, the latent representation z generated from the input sentence x through VAE will be adjusted before sending to the decoder. In detail, the latent vector z will be added the target style embedding and subtracted the style embedding of original style as x. Intuitively, we expect the injection and removal of style information is completed by the addition and subtraction operations of style embeddings. The updated latent representation is expressed as follows:

$$z' = z + w \cdot (s_t - s_o) \quad (7)$$

where s_t and s_o are target and original style embeddings trained in the stage I respectively. w represents the style weight that adjusts the style strength applied to the sentence generation. A higher weight means stronger style attributes will be injected for generation.

3 Experiments

3.1 TST Evaluation Tasks and the Settings

We evaluate VT-STOWER with three different TST tasks: sentiment transfer, formality transfer, and code-switching transfer.

For sentiment transfer, we adopt the Yelp dataset (Li et al., 2018), in which each sample is a business review of a restaurant and is labeled as positive or negative. For formality transfer, We adopt one of the largest corpus for formality transfer task, namely *Family and Relationships* domain data in GYAFC (Grammarly's Yahoo Answers Formality Corpus) (Rao and Tetreault, 2018). For code-switching transfer, we evaluate VT-STOWER on a Hindi→Hinglish transfer task, which is extracted from the English-Hinglish translation dataset at LinCE (Linguistic Code-switching Evaluation) (Aguilar et al., 2020). We first translate English sentences into Hindi by Amazon Translation Service and then transliterate Latin scripts of Hindi words into Devanagari form by using *indic-trans* tool (Bhat et al., 2014) to keep the consistency of the script of language [2]. Note this dataset is very low-resource, which only contains 7K sentences for training. Similar to GYAFC set, we shuffle the training data and treat it as unpaired data. Note that two of the training sets are transformed from originally paired dataset instead of directly using richer unpaired datasets, it is because we want to make a fair comparison with other referenced approaches. The training and test set size for each task is presented in Table 1.

Tasks	training set	evaluation set	test set
Sentiment Transfer (positive/negative)	266K/177K	2K/2K	500/500
Formality Transfer (formal/informal)	52K/52K	2.2K/2.7K	1K/1.3K
Code-Switching Transfer (Hinglish/English)	7K/7K	300/300	300/300

Table 1: Training, evaluation, and test set size of three evaluation tasks.

Considering that GYAFC and Yelp dataset are written in English and the code-switching dataset is in mixed languages of Hindi and English, we use `RoBERTa` as the pre-trained language model for sentiment and formality transfer tasks, and `XLM-R` (Conneau et al., 2020) for code-switching transfer. Also, we fine-tune the style classifier to obtain important score distribution by leveraging `RoBERTa` and `XLM-R` for the corresponding transfer tasks.

[2]The output of translator is Devanagari form while the original script of Hindi in LinCE is Latin.

More training hyperparameters are shown in Appendix A.

3.2 Evaluation Metrics

Style Transfer Accuracy (Acc) Style transfer accuracy (Acc) is defined as the ratio of the number of successfully transferred sentences and the total number of input sentences. Following previous studies (Dai et al., 2019; Sudhakar et al., 2019), we leverage fastText classifier (Joulin et al., 2017) to classify whether the original text have been successfully transferred to the target style. The classifier is trained on the same training data used for style transfer. The three classifiers achieve 97.6%, 85.75% and 99.7% accuracy for sentiment, formality and code-switching style classification itself, respectively.

Perplexity (PPL) We also measure the fluency of the transferred sentences by calculating their perplexity. The lower the perplexity is, the more fluent the generated sentences are. For the GYAFC and Yelp dataset which are in English, we use the pre-trained language model GPT2 (Radford et al., 2019) to compute the perplexity, where no further fine-tuning is conducted. However, GPT2 does not apply to other languages or code-switching sentences. Following Samanta et al. (2019), we train a character-level LSTM (Kim et al., 2016) on the code-switching training data and utilize this model to derive the perplexity of generated code-switching sentences.

BLEU Scores Content preservation is evaluated by the tokenized BLEU scores (Papineni et al., 2002) between the transferred sentences and human-authored references, which is calculated with the `multi-bleu.perl`. Note that GYAFC dataset has four human references, so the BLEU for GYAFC is the mean BLEU scores between the generated sentences and four references. Because there is no human reference for code-switching task, we report BLEU scores between transferred sentences and original sentences for code-switching transfer instead.

Geometric Mean (GM) Following Yi et al. (2020), We also report the geometric mean of accuracy, BLEU, $\frac{1}{\ln \text{PPL}}$ as the overall performance.

3.3 Main Results

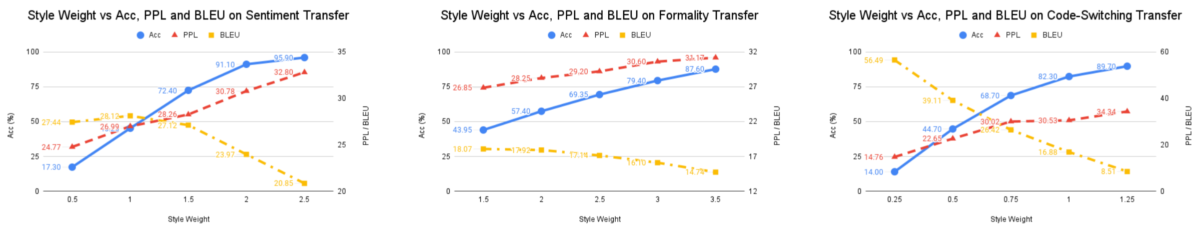
The performance of VT-STOWER and previous works are shown in Table 2. First of all, we can

Models	Acc ↑	PPL ↓	BLEU ↑	GM ↑
Sentiment Transfer (Yelp)				
CrossAlignment	74.0	42.91	9.06	5.63
Delete & Retrieve	87.5	40.66	5.99	5.21
B-GST	84.3	**25.27**	22.82	8.41
Style Transformer	83.9	43.60	**28.29**	8.57
Deep LatentSeq	83.0	27.08	24.03	8.46
StyIns	91.5	42.60	25.11	8.49
Tag & Generate	87.5	32.98	21.80	8.17
Ours (stage I, $w = 4$)	**91.7**	38.35	18.51	7.75
Ours (stage II, $w = 2$)	91.1	30.78	23.97	**8.61**
Human Reference	74.1	27.40	100.0	13.08
Formality Transfer (GYAFC)				
CrossAlignment	65.35	**13.66**	1.57	3.40
Delete & Retrieve	53.85	29.70	11.71	5.71
Style Transformer	56.05	48.72	**24.67**	7.09
Ours (stage I, $w = 4$)	80.9	31.90	14.19	6.92
Ours (stage II, $w = 3.1$)	**81.0**	30.78	15.84	**7.21**
Human Reference	82.31	28.05	100.0	13.39
Code-Switching Transfer (LinCE)				
Style Transformer	**99.3**	601.45	3.47	3.78
Randomly Replace	1.02	213.24	**69.09**	2.36
Ours (stage I, $w = 0.75$)	66.67	29.91	24.30	7.81
Ours (stage II, $w = 0.75$)	68.70	30.02	26.42	**8.11**

Table 2: Overall results of our models (VT-STOWER) and previous methods on three style transfer tasks. The best scores are bolded in the corresponding metric. ↑ means the higher is better, vice versa.

clearly see the performance improvement brought by stage II training compared with single training stage I. **In all three transfer tasks, models trained in stage II lead to lower PPL (or similar PPL in code-switching transfer) and higher BLEU scores when we find a w to control them in a similar Acc, which achieves better overall performance.** For instance, compared with the model trained in stage I with $w = 4$ in the sentiment transfer, the model fine-tuned in stage II achieves similar accuracy with $w = 2$ (91.7% vs. 91.1%). At the same time, the stage II model decreases the PPL from 38.35 to 30.78 and increases the BLEU from 18.51 to 23.97, which demonstrates that the pivot words masking training is capable of improving the smoothness of the sentences and the preserving the content. Note that we cannot use the same weight w for a direct comparison since the models in two training stages have different sensitivity to w. Therefore, we use w that produces similar accuracy between stage I and stage II for a fair comparison for the PPL and BLUE.

When comparing our method with several state-of-the-art references: CrossAlignment (Shen et al., 2017), Delete & Retrieve (Li et al., 2018), B-GST (Sudhakar et al., 2019), Style Transformer (Dai et al., 2019), Deep LatentSeq (He et al., 2020), Tag & Generate (Madaan et al., 2020) and StyIns



(a) sentiment transfer

(b) formality transfer

(c) code-switching transfer

Figure 3: Illustration of style weight w vs. Acc, PPL and BLEU in sentiment, formality and code-switching transfer tasks. Note there is a trade-off between Acc and PPL/BLEU. With increasing of w, Acc will increase while BLEU drops down and PPL increases.

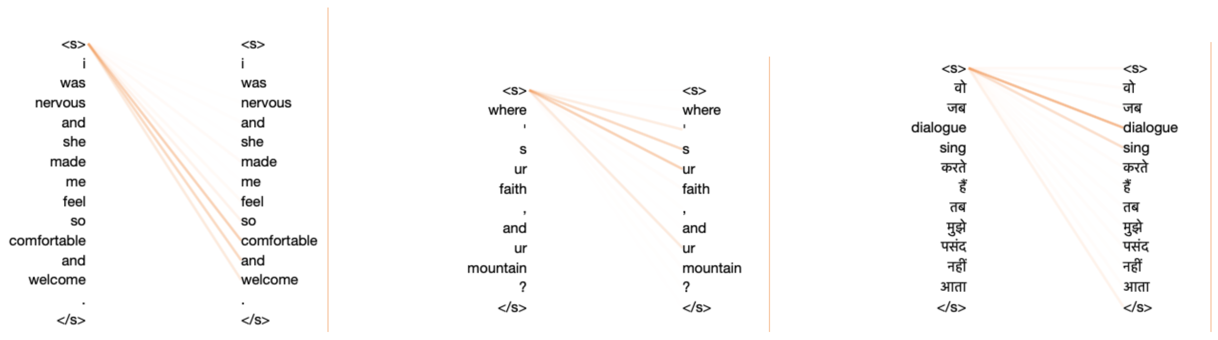
	Negative → Positive Transfer (Yelp)
Original	the place has obviously gone downhill over the years .
Style Transformer	the place has consistently gone handful over the years .
VT-STOWER	the place has greatly improved over the years .
	Informal → Formal (GYAFC)
Original	ask her out or tell her u like or admire her .
Style Transformer	ask her out or tell her is like or admire her .
VT-STOWER	ask her out or inform her that you like her .
	Code-Switching Transfer (LinCE)
Original	म सहमत ह ! डालता ह । मझ यकीन नही था कि अत म कस महसस किया जाए । (I agree! I was not sure how to feel so.)
Style Transformer	movie की हैं (are movie)
Randomly Replace	म सहमत ह ! डालता ह । मझ यकीन no had कि अत म tight महसस किया जाए । (I agree! I put I sure had no idea that I could be felt so tight.)
VT-STOWER	i agree ! मैं sure नहीं था की end मैं कैसा feel करू (i agree ! I wasn't sure how would I feel)

Figure 4: Comparison of style transfer outputs of our models and style transformer in three transfer tasks. Our models are stage II models in Table 2. Translations for code-switching sentences are shown in parenthesis.

(Yi et al., 2020), the performance of their methods is directly evaluated on their provided outputs by using our metric evaluators. We will further discuss how w affect the performance in next section. We can clearly see the overall performance (GM) of our proposed model is better than all baselines. For evaluating the success of style transfer, accuracy is the most critical metrics, for which VT-STOWER also demonstrates large improvement in sentiment and formality transfer.

In the sentiment style transfer, our model with $w = 2$ (after stage II training) has competitive accuracy (91.1%), and BLEU (23.97) compared with the state-of-the-art methods StyIns (Yi et al., 2020) (accuracy=91.5%, BLEU=25.41) and style transformer (Dai et al., 2019) (accuracy=83.9%, BLEU=28.29) but achieve much lower perplexity (30.78) compared to 42.60 in StyIns and 43.60 in style transformer, which demonstrates that the sentences generated from our model is closer to the natural language. VT-STOWER also outperforms other previous methods by a large margin in all three metrics. In the formality transfer, the most competitive model is the style transformer. Although it achieves higher BLEU scores (24.67), our models beats it on higher style transfer accuracy (81.0% vs. 56.05%) and significantly lower PPL (30.78 vs. 48.72) with limited loss of BLEU scores.

For the code-switching transfer, since there is no previous TST experimenting on this task, we train the strongest baseline (style transformer) for this task. Interestingly, style transformer obtains a very high accuracy (99.8%) with the costs of very high PPL (601.46) and very low BLEU score (3.47). The possible reason is that the style transformer is only able to capture partial special style features from the small dataset (7K) and only transfer sentences based on these features without fully capturing the nature of languages, resulting in high accuracy but low fluency and BLEU. However, VT-STOWER can balance among the accuracy, fluency, and BLEU to achieve reasonable results even in the case of the low-resource dataset, which demonstrates its generalization power. Additionally, we also design another baseline, i.e., we randomly replace 15% Hindi words with English words (Zheng et al., 2021) based on the MUSE dictionary (Conneau et al., 2017), because intuitively, people may regard code-switching text generation as simply translating several words. However, this method only achieve 1.02% accuracy, because simple translation and replacement cannot accord with the habit of bilingual expression in code-switching sentences, namely, code-switching has its own style according to the speakers (e.g., usually noun is more likely to be replaced with foreign language than preposition). Intuitively, when we compare the VT-STOWER with original and other approaches as shown in Figure 4, the output bilingual sentence from VT-STOWER reads more fluent and can be easier understood.



<s> i was nervous and she made me feel so comfortable and welcome . </s>
<s> i was nervous and she made me feel so comfortable and welcome . </s>
<s> where ' s ur faith , and ur mountain ? </s>
<s> where ' s ur faith , and ur mountain ? </s>
<s> वो जब dialogue sing करते हैं तब मुझे पसंद नहीं आता </s>
<s> वो जब dialogue sing करते हैं तब मुझे पसंद नहीं आता </s>

(a) sentiment transfer (b) formality transfer (c) code-switching transfer

Figure 5: Illustration of the mean attention weights of token '<s>' from all heads at the final layer in three TST tasks. Higher importance scores are assigned to pivot words, which are depicted as deeper lines in the figures.

3.4 Effect of Style Weights

As shown in Equation 7, the strength of the target style in z' is adjusted by the style weight w. In Figure 3, we present metrics trend with five different w for the models trained in stage II [3], and demonstrate how the style weight w affects the outputs. Taking sentiment transfer task as an example, when w is increased from 0.5 to 2.5, the transfer accuracy climbs from 17.3% up to 95.9%, but the BLEU score drops from 27.44 down to 20.85, and PPL increases from 24.77 to 32.8. The reason is when increasing w, more style information is injected into the latent vector that the decoder pays more attention to the target style feature rather than the naturalness and content of generated sentences. Therefore, w is a trade-off hyperparameter between the transfer accuracy and PPL/BLEU. Examples of generated sentences transferred from positive to negative sentiment with $w = 1.5, 2, 2.5$ are illustrated in Table 3. When $w = 1.5$, the model still can find a positive word, 'enjoying,' which makes the sentence ironical. In the case of $w = 2$, the 'enjoying' is rephrased to 'avoid', turning the sentence into a full negative attitude. If we further increment $w = 2.5$, more negative words will be added regardless of the smoothness of the sentence. Similar discussions also hold for the formality and code-switching transfer, where their results versus various style weights are illustrated in Figure 3b and 3c.

3.5 Importance Score Distribution

Recall that for the training stage II, the importance scores are derived from the attention scores of the BOS token '<s>', which are the mean scores of all heads from the last layer of a pre-trained encoder. Figure 5 presents the examples of importance score, showing how the score value represents the importance of words in terms of style representation. The importance scores are higher on 'comfortable and welcome' in the sentiment transfer, these words represents strong positive emotions. Similarly, the scores are higher on the informal written words 'ur' in the formality transfer, and English words mixed in a Hinglish sentence in the code-switching transfer.

Positive → Negative with various w	
Original	i will be going back and enjoying this great place !
$w = 1.5$	i will be going back and enjoying this **terrible** place !
$w = 2$	i will be going back and **avoid** this **terrible** place !
$w = 2.5$	i will be going back and **worst rude avoid** this **terrible** place !

Table 3: Examples of sentences transferred from positive to negative sentiment with various settings of w. The higher w is the more negative words are injected in the sentences.

4 Conclusion

We proposed the VT-STOWER, a model joinly trained with VAE and style embeddings for content distribution and style information. The method successfully transfers several different text styles, including the code-switching TST task for the first time. Taking advantage of the flexibility of style embeddings, our proposed model has the ability to adjust the style strength during the transfer by simply adjusting the style weights. To further enhance the transfer accuracy, we propose additional pivot words masking training scheme, which shows impressive improvement.

[3] w ranges from 0.5 to 2.5 with an interval of 0.5 for sentiment and formality transfer, and from 0.25 to 1.25 with an interval of 0.25 for code-switching transfer.

References

Gustavo Aguilar, Sudipta Kar, and Thamar Solorio. 2020. LinCE: A Centralized Benchmark for Linguistic Code-switching Evaluation. In Proceedings of The 12th Language Resources and Evaluation Conference, pages 1803–1813, Marseille, France. European Language Resources Association.

Irshad Ahmad Bhat, Vandan Mujadia, Aniruddha Tammewar, Riyaz Ahmad Bhat, and Manish Shrivastava. 2014. Iiit-h system submission for fire2014 shared task on transliterated search. In Proceedings of the Forum for Information Retrieval Evaluation, FIRE '14, page 48–53, New York, NY, USA. Association for Computing Machinery.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning, pages 10–21, Berlin, Germany. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Édouard 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.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. arXiv preprint arXiv:1710.04087.

Ning Dai, Jianze Liang, Xipeng Qiu, and Xuanjing Huang. 2019. Style transformer: Unpaired text style transfer without disentangled latent representation. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 5997–6007, Florence, Italy. Association for Computational Linguistics.

Yao Fu, Hao Zhou, Jiaze Chen, and Lei Li. 2019. Rethinking text attribute transfer: A lexical analysis. In Proceedings of the 12th International Conference on Natural Language Generation, pages 24–33.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 32.

Junxian He, Xinyi Wang, Graham Neubig, and Taylor Berg-Kirkpatrick. 2020. A probabilistic formulation of unsupervised text style transfer. In International Conference on Learning Representations.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2016. beta-vae: Learning basic visual concepts with a constrained variational framework.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In International Conference on Machine Learning, pages 1587–1596. PMLR.

Di Jin, Zhijing Jin, Zhiting Hu, Olga Vechtomova, and Rada Mihalcea. 2020. Deep learning for text style transfer: A survey. arXiv preprint arXiv:2011.00416.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2019. Disentangled representation learning for non-parallel text style transfer. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 424–434, Florence, Italy. Association for Computational Linguistics.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In Thirtieth AAAI conference on artificial intelligence.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. ICLR.

Guillaume Lample, Sandeep Subramanian, Eric Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2019. Multiple-attribute text rewriting. In International Conference on Learning Representations.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. 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 1865–1874, New Orleans, Louisiana. 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. arXiv preprint arXiv:1907.11692.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Rätsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. 2019. Challenging common assumptions in the unsupervised learning of disentangled representations. In ICML.

Aman Madaan, Amrith Setlur, Tanmay Parekh, Barnabas Poczos, Graham Neubig, Yiming Yang, Ruslan Salakhutdinov, Alan W Black, and Shrimai Prabhumoye. 2020. Politeness transfer: A tag and generate approach. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1869–1881, Online. 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 of the Association for Computational Linguistics, pages 311–318.

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.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. OpenAI blog, 1(8):9.

Sudha Rao and Joel Tetreault. 2018. Dear sir or madam, may I introduce the GYAFC dataset: Corpus, benchmarks and metrics for formality style transfer. 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 129–140, New Orleans, Louisiana. Association for Computational Linguistics.

Bidisha Samanta, Sharmila Reddy, Hussain Jagirdar, Niloy Ganguly, and Soumen Chakrabarti. 2019. A deep generative model for code-switched text. IJCAI.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment.

Akhilesh Sudhakar, Bhargav Upadhyay, and Arjun Maheswaran. 2019. “transforming” delete, retrieve, generate approach for controlled text style transfer. 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 3269–3279.

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.

Xiaoyuan Yi, Zhenghao Liu, Wenhao Li, and Maosong Sun. 2020. Text style transfer via learning style instance supported latent space. In Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, pages 3801–3807. International Joint Conferences on Artificial Intelligence Organization. Main track.

Bo Zheng, Li Dong, Shaohan Huang, Wenhui Wang, Zewen Chi, Saksham Singhal, Wanxiang Che, Ting Liu, Xia Song, and Furu Wei. 2021. Consistency regularization for cross-lingual fine-tuning. arXiv preprint arXiv:2106.08226.

A Hyperparamerters

The encoder and decoder for sentiment and formality transfer tasks both are two-layer transformer (Vaswani et al., 2017), with FFN dimension size of 1024 and 4 attention heads. Due to the limited code-switching data size, we run smaller encoder and decoder with 256 FFN dimensions and 2 attention heads for code-switching transfer task. The neural networks that formulate the mean and variance of the latent space are also one-layer transformer blocks. The dimension of the latent features and style embeddings is 768. Both penalty weights λ_{vae} and λ_{style} equal to 1. We set γ as 0.01, 0.03, and 0.005 for sentiment, formality, code-switching transfer during importance score calculation, respectively. The β is set as 1. The optimizer is Adam (Kingma and Ba, 2014) with learning rate 0.0005. The batch size is 8092 tokens.

MRE : Multi Relationship Extractor for Persona based Empathetic Conversational Model

Bharatram Natarajan
SRIB*
bharatram.nitt@gmail.com

Abhijit A Nargund
SRIB*
nargundabhi@gmail.com

Abstract

Artificial intelligence(AI) has come a long way in aiding the user requirements in many fields and domains. However, the current AI systems do not generate human- like response for user query.Research in these areas have started gaining traction recently with explorations on persona or empathy based response selection. But the combination of both the parameters in an open domain haven't been explored in detail by the research community. The current work highlights the effect of persona on empathetic response. This research paper concentrates on improving the response selection model for PEC dataset, containing both persona information and empathetic response. This is achieved using an enhanced **multi relationship extractor** and **phrase based information** for the selection of response.

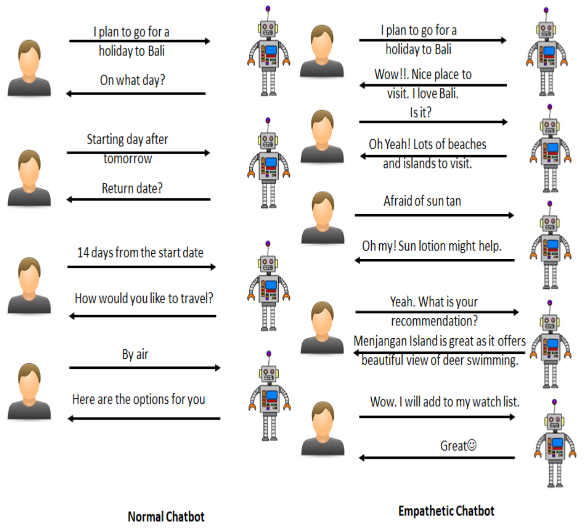

I plan to go for a holiday to Bali
On what day?
Starting day after tomorrow
Return date?
14 days from the start date
How would you like to travel?
By air
Here are the options for you
I plan to go for a holiday to Bali
Wow!!. Nice place to visit. I love Bali.
Is it?
Oh Yeah! Lots of beaches and islands to visit.
Afraid of sun tan
Oh my! Sun lotion might help.
Yeah. What is your recommendation?
Menjangan Island is great as it offers beautiful view of deer swimming.
Wow. I will add to my watch list.
Great☺
Normal Chatbot
Empathetic Chatbot

Figure 1: Normal Chatbot Vs Empathetic Chatbot

1 Introduction

Empathetic response generation refers to the ability of a system to understand the people mentality or the feelings of a user and provide an appropriate response. In the area of NLP, empathetic conversational models have shown a positive impact on the user compared to normal responses[Liu and Picard (2005);Zhou et al. (2020);Lin et al. (2020);Li et al. (2021)].Empathetic response creates a personal connect with the user and achieves significantly higher engagement by providing seamless conversational experience as shown by this example 1. You can see how empathetic response aides in different kind of engagement with the user and also provide suggestions.

Recent years have seen an increase in exploration on empathetic response generations using neural conversational models.Lin et al. (2020) have developed CAIRE to generate empathetic response using user emotions for better connections with the user.Li et al. (2021) have used causal emotional information to understand a particular user emotional state and correspondingly learn response for each emotion class. Zhou et al. (2020) have developed an AI model named XiaoIce for better human understanding and communication. (Rashkin et al., 2018) has created custom datasets for the said problems named EMPATHETICDIALOGUES (ED). These dataset has proven that pre-trained retrieval models like BERT and its variants are able to reply with more empathy when trained with such dataset.

The above explorations doesn't take into account the persona of the user.As persona aides in better conversational response, this area started gaining traction by the research community.Demasi et al. (2020) developed Crisisbot where it uses the persona of the user in the conversation to provide complex response to train hotline counselors for suicide prevention task.Song et al. (2019) explored the way to generate sustainable and coherent response, using persona in a conversation, where each request will have many possible responses.Wang et al.

*Samsung R&D Institute India - Bangalore

(2021) developed an emotion-affective open domain chatbot where they use knowledge graph to extract personal information and incorporate into the system for consistent personality.

The presence of Persona have shown impact in the conversational response. Hence, exploration on the impact of persona on empathetic response started gaining traction. Persona have shown to impact empathetic natural language generation capability. We have noticed that empathetic response of system is different from two different users for the same input user utterance or query.Roller et al. (2020) incorporated persona with empathy on response generation for several dataset but the impact of persona on empathy is not explored. Zhong et al. (2020) has presented a novel large-scale multi-turn Persona-based Empathetic Conversation (PEC) using two contrasting sentimental domains from social media Reddit. They have proposed novel Cobert architecture (combination of BERT with basic Co-Attention mechanism) to select the appropriate response for the post and penalise opposite response pairs. This is the first approach where Persona information is used to influence the empathetic response selection. The current architecture takes the full user context, personas and response for gathering bi-attention and selecting appropriate response. Co-attention won't be able to capture phrase level impact on the response selection. Moreover the influence of the different phrases from different positions might not be captured well.

To address this issue, we are proposing multi relationship extraction using BERT for influencing the selection of appropriate response and penalising the un-important ones.

In summary, the contributions of the papers is summarized as below

- We propose phrase importance planner to extract n-gram impact of the phrase in unigram.
- We propose multi relationship extraction using BERT where we use phrase importance planner along with the entire context to impact response.

2 Related Work

2.1 Retrieval based Conversational Model

Lots of work have happened in neural conversational model for response selection task. The task is approached in 3 stages which are as shown below.

- **Encoding** The encoding module encodes the input tokens into contextual vector representation using encoders like BERT, ELMO or non contextual vector representations like Glove embedding.
- **Matching** The matching module measures the co-relation between user context and response using persona details with different attention techniques.
- **Aggregation** The aggregation module aides in summarising the matching module information along the sequence axis to get the final representation.

Humeau et al. (2019) introduced polyencoders for handling pairwise comparison between 2 sequences using the combination of cross encoders and bi-encoders.Cross-encoders calculates attention over all the target labels and hence will be slow in calculation. Bi-encoders calculate individual pair attention and will be faster.

3 PEC Dataset

This section explains the high level information on PEC dataset gathered by Zhong et al. (2020). For full detailed analysis, please refer to the above paper. PEC dataset is available in huggingface as [1].

Data Source The data has been collected by the author Zhong et al. (2020) from Reddit, a discussion forum where users can discuss any topics on sub-reddits. The data is made from two contrasting sentiments related sub-reddits namely **Happy** and **Offmychest**. Here the comments are more empathetic than casual conversations.

Conversation Collection Reddit has threads where there will be a single post containing direct and indirect comments. These threads are organized in a tree where the root represent post and comments are represented as nodes connecting to parent comment node or root post node. If there are n nodes then author extracts n-1 conversations where each conversation starts from root node and ends at n-1 non root nodes. The author has split the data into 80:10:10 for training, validation and testing.

Persona Collection Persona sentences are collected from all the posts and comments that the

[1]https://huggingface.co/datasets/pec

user has written. There are strict rules applied to fetch the persona sentences from the provided posts which are listed below

- Presence of the word ”i” in the post.
- Presence of atleast one verb
- Presence of atleast one noun or adjective.
- Presence of atleast one content word.

Data Processing We follow the data processing steps followed by the author Zhong et al. (2020). These steps are listed below.

- Each conversation has at most 6 most recent turns.
- Each post is between 2 and 90 words.
- Each comment is between 2 and 30 words.
- Each speaker has atleast one persona sentence.
- The last speaker is different from the first speaker in each conversation. The reason is last comment is considered as empathetic response rather than reply to the user.
- Remove all special symbols, URLs and image captions.
- Lowercase all the utterances.

Sample conversation of PEC dataset for happy and offmychest are present in Table 1

Conversations	
OffMyChest	**Happy**
why is it ok for women to wear skirts in a business casual environment , but men ca n't wear shorts ?	got the best t - shirt ever today !
skirts generally are n't comfortable . you ca n't do much in them other than walk , unless they 're long and even then ...	prepare your inbox for pussy puns
my issue is n't comfort , it 's sweating my balls off . ladies also get to wear sleeveless shirts !	i do n't get it ..
... not sweating your balls off is comfort .	she 's a girl wearing a shirt with a cat on it . you know what , no ... this has got ta be a troll attempt
Persona of respondent	
i was going to say ” welcome to being a woman ”	i have 2 characters .
i just wanted to share .	i respond the same way to gal gadot tickling me
i make lots of male friends easy .	i know that feel , unfortunately

Table 1: Sample data for **OffMyChest** and **Happy** classes in the PEC dataset.First 4 rows represent Conversation and last 3 rows represent Persona details of the respondent.

4 Multi Relationship Extractor using BERT Embedding

This section introduces the task of response selection and briefly explains novel architecture on addressing the task at hand as shown in Figure 2.

4.1 Task Definition

We denote the training conversational dataset as D_X. D_X is a set of n conversations. Each conversation is in the format of (U_X, P_X, Y_X) where $U_X = U_X1, U_X2, U_X3...,U_Xn$ indicates the n user context utterances, $P_X = P_X1, P_X2, P_X3, ..., P_Xp$ denotes the p persona sentences for the respondent and Y denotes the target response for the user context. We can formulate the response selection problem as f(U_X, P_X, Y_X) where we assign highest probability to true candidate Y_X and lowers the score of negative candidates given X and P. When we infer the model, the model will select the best candidate from the candidate list by selecting highest probability.

4.2 BERT Embedding

This module handles the first stage of retrieval model namely Embedding step. In this module, we encode the user context utterances, persona utterances and response utterances using BERT pretrained model (Devlin et al., 2018). User context utterances is obtained by concatenating the list of sentences uttered by the user in order. Persona utterances are obtained by random ordered concatenation of list of persona utterances for the respondent. Response utterances are obtained by concatenating the list of response sentences. When we encode context utterances, persona utterances and response utterances using BERT, we get vector representation of context, persona and response. Context

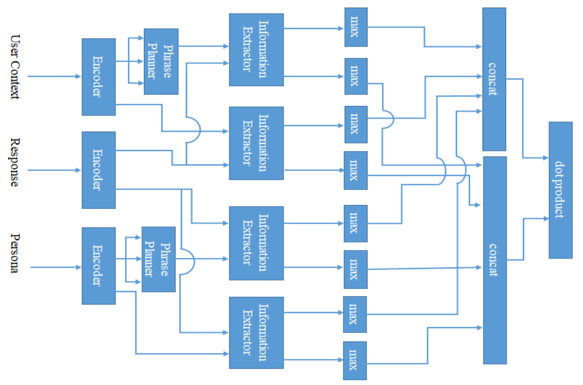


Figure 2: Multi Relationship Extractor using BERT.Information Extractor module address the importance of phrase or context level learning over response and vice versa. Phrase Planner handles phrase importance of user context or persona inputs by incorporating the importance of bigram and trigram over unigram.

vector representation will be C $\epsilon \mathbb{R}^{c \times d}$ where c is the maximum sequence length of the user context utterances.Persona vector representation will be P $\epsilon \mathbb{R}^{p \times d}$ where p is the maximum sequence length of the persona utterances. Response vector representation will be R $\epsilon \mathbb{R}^{r \times d}$ where r is the maximum sequence length of the response utterances. One important information is that we use different segment ids for user context utterances and response utterances. Now Context vector representation and Persona vector representation will pass through Phrase Planner Module.

4.3 Phrase Planner Module

This submodule aides in capturing the phrase importance for both Context vector and Persona vector separately as shown in Figure 3. For Context vector, this is done by capturing unigram, bigram and trigram information using 3 different 2d convolution module with sliding window size of 1, 2 and 3 tokens respectively keeping padding same. The importance of bigram and trigram on unigram is captured using Multi Head Attention(MHA) module developed by Vaswani et al. (2017) as shown in Figure 4. We use separate MHA module for calculating different positional bigram importance on unigram and different positional trigram importance on unigram. MHA achieves this by dividing the Query(Q), Key(K), Value (V) input into equal chunks and process each chunk in parallel. Each chunk calculates the weights for V using scaled dot product followed by softmax as shown in Equation 1. The scalar dot product calculates the affinity or importance of Query on Key.Now we concatenate each chunk output at last dimension layer to get final matrix.

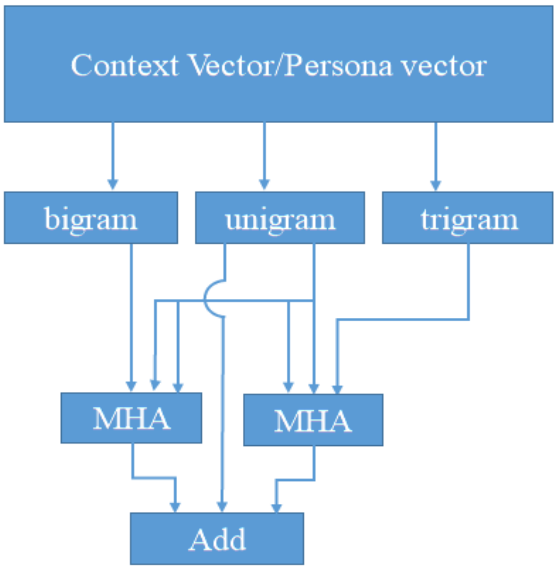


Figure 3: Phrase Planner Module.

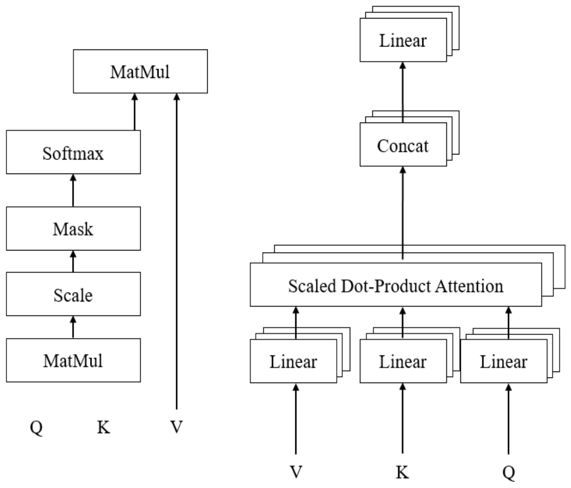


Figure 4: Multi Head Attention.

$$Attention(Q, K, V) = softmax(QK^T/\sqrt{d})V \quad (1)$$

For bigram, we pass the Query as bigram, Key as unigram and Value as unigram in this module. Similarly we calculate for Trigram importance in unigram. Finally we add unigram, bigram importance and trigram importance modules to get final matrix named Context phrase Planner.This module maintains the dimension of the input vector. Hence the output for Context phrase Planner is $C^{PP} \epsilon \mathbb{R}^{c \times d}$. Similarly we apply different Phrase Planner Module for Persona vector and generate Persona phrase planner as $P^{PP} \epsilon \mathbb{R}^{p \times d}$. Now we pass the Context phrase Planner,Persona phrase planner, Context vector, Persona vector and response vector will be passed through Information Extractor module.

4.4 Information Extractor

This module is responsible for mutual importance relations projection between the following learnings.

- Context vector C and Response vector R.
- Context Phrase Planner C^{PP} and Response vector R.
- Persona vector P and Response vector R.
- Persona Phrase Planner P^{PP} and Response vector R.

This module handles the second stage of retrieval model namely Matching step.Sample flow of Information Extractor module is shown in Figure 5 For a given C^{PP} and R, we are calculating

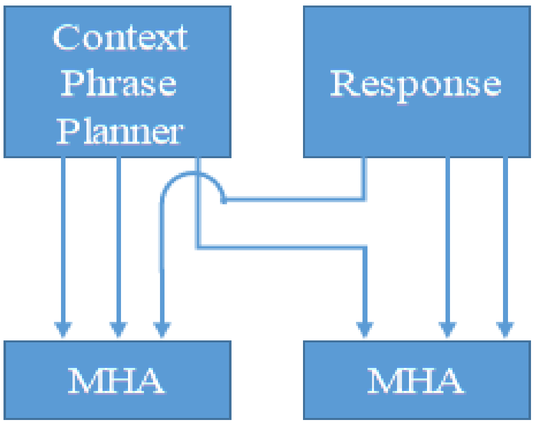


Figure 5: Mutual Information Projector contains 2 inter relations MHA where 1 MHA calculates the importance of Context phrase Planner to response and other calculates the importance of Response to Context phrase Planner.

the importance of C^{PP} that should be projected to R and the importance of R that should be incorporated on C^{PP}. This is done with the help of 2 Multi Head Attention modules. The first Multi Head attention module calculates the importance of C^{PP} on R by passing Query as C^{PP}, Key as R and Value as R. The scale dot product attention, followed by softmax calculates the affinity of C^{PP} and R to create matrix $\epsilon \mathbb{R}^{p \times r}$.The weighted affinity matrix will be multiplied with Value matrix to get impact of C^{PP} on R for $R_{PP} \epsilon \mathbb{R}^{p \times d}$. Similarly we use another Multi head Attention to calculate the importance of R on C^{PP}. Here we use Query as R, Key as C^{PP} and Value as C^{PP}.The scale dot product attention, followed by softmax calculates the affinity of C^{PP} and R to create matrix $\epsilon \mathbb{R}^{r \times p}$.The weighted affinity matrix will be multiplied with Value matrix to get impact of R on C^{PP} as $C_R^{PP} \epsilon \mathbb{R}^{r \times d}$. The same is applied for Context vector C and Response vector R, Persona vector P and Response vector R and Persona Phrase Planner and Response vector R to get $C_R \epsilon \mathbb{R}^{r \times d}, R_C \epsilon \mathbb{R}^{c \times d}, P_R \epsilon \mathbb{R}^{r \times d}$, $R_P \epsilon \mathbb{R}^{p \times d}, R_{PPP} \epsilon \mathbb{R}^{p \times d} and P_R^{PP} \epsilon \mathbb{R}^{r \times d}$. All these learnings are passed through max pooling layer which takes the maximum along the sequence dimension to generate $R_{CPP_{max}} \epsilon \mathbb{R}^d, C_{Rmax}^{PP} \epsilon \mathbb{R}^d$, $C_{Rmax} \epsilon \mathbb{R}^d, R_{Cmax} \epsilon \mathbb{R}^d, R_{PPP_{max}} \epsilon \mathbb{R}^d, P_{Rmax}^{PP} \epsilon \mathbb{R}^d$, $P_{Rmax} \epsilon \mathbb{R}^d, R_{Pmax} \epsilon \mathbb{R}^d$.

4.5 Dot Product

We concatenate the response importance learnings as $R_f = R_{CPP_{max}}; R_{Cmax}; R_{PPP_{max}}; R_{Pmax} \epsilon \mathbb{R}^{4d}$ and $U_{xf} = C_{Rmax}^{PP}; C_{Rmax}; P_{Rmax}^{PP}; P_{Rmax} \epsilon \mathbb{R}^{4d}$. Then we use dot product to calculate final matching score as shown in equation 2.

$$f(U_X, P_X, Y_X) = dot(R_f, U_{xf}) \qquad (2)$$

Model is optimized by reducing the cross entropy loss between target true candidate and final matching score.

5 Experiments

This section explains the baseline models, experimentation and model comparisons.

5.1 Baseline Models

We compare our models with BoW, HLSTM, Bi-encoder, Co-BERT for PEC dataset.

- **BoW**: tri-encoder architecture with average word embedding for context, response and persona.
- **HLSTM**: Makes use of utterance level Bi-LSTM and context level Bi-LSTM. Also all encoders share same utterance level Bi-LSTM.

	Happy				OffMyChest				All			
Models	**R@1**	**R@10**	**R@50**	**MRR**	**R@1**	**R@10**	**R@50**	**MRR**	**R@1**	**R@10**	**R@50**	**MRR**
BoW	10.2	45.6	85.2	21.8	13.9	51.6	87.1	26.2	15.4	52.9	86.7	27.4
HLSTM	15.7	53.6	91.6	28.1	17.6	55.7	91.8	30.2	22.2	63.0	94.8	35.2
DIM	31.3	67.0	95.5	43.0	40.6	72.6	96.4	51.2	39.3	74.6	97.3	50.5
Bi-encoder	32.4	71.3	96.5	45.1	42.4	78.4	97.6	54.5	42.4	78.4	97.6	54.5
Poly-encoder	33.7	72.1	96.7	46.4	43.4	79.3	97.7	55.3	42.3	79.2	98.1	54.4
Co-Bert	36.2	73.0	96.9	48.4	47.0	79.7	97.8	58.0	45.1	80.5	98.3	56.7
Our Model	**37.8**	**75.2**	**97.5**	**49.6**	**48.1**	**81.2**	**98.3**	**59.4**	**46.2**	**81.2**	**98.7**	**57.4**

Table 2: Comparison of state of the art models for Happy, OffMyChest and All

- **DIM**: Makes use of fine grained matching and hierarchical aggregation to learn rich matching information.
- **Bi-encoder**: State of the art BERT based model for empathetic response selection.
- **PolyEncoder**: gains an understanding of latent attention codes for finer grained matching.

5.2 Evaluation Metrics

We follow the same evaluation metrics proposed by Zhong et al. (2020). We evaluate the models on Recall@k where k candidates needs to be selected from C samples. We abbreviate it as R@k. We use k as 1, 20 and 50. We use C as 100.We also measure Mean reciprocal Rank (MRR). MRR calculates the mean of reciprocal of the rank of the correct response. The rank of the correct response is calculated by finding the position of the correct response id inside the list of predicted response ids sorted in decreasing order by probabilities.

5.3 Baseline Comparison

Table 2 shows the experimentation results of the models for PEC dataset namely Happy, Offmychest and All.

From the above table, we are able to observe that sentence representation is one of the most important critical factor for response selection. Another important factor that is noticebale is the fine grained matching logic which aides in better response selection. Sentence representation information importance is visible between BoW, HLSTM, DIM and Bi-encoder where Bi-encoder model has outperformed other models. CoBert has performed best amongst all the other models (except our model) mainly because of first-order and second-order multi-hop co-attention calculation which aided in better response, user context pair with the help of persona. Our model is able to defeat Cobert model because of additional phrase level projection of user context and persona on response. In addition, the mutual information extractor module aided in better relationship between persona, user context to response which enhanced the response selection.

6 Conclusion

We are able to observe that additional phrase level information flow, both for user context as well as persona, aided in better relationship building between response and context as well response and persona which in turn aided in better response selection. In addition the Multi head attention aided in multi phrase positional information capture which resulted in better learning representation.

References

Orianna Demasi, Yu Li, and Zhou Yu. 2020. A multi-persona chatbot for hotline counselor training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 3623–3636.

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*.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2019. Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring. *arXiv preprint arXiv:1905.01969*.

Yanran Li, Ke Li, Hongke Ning, Xiaoqiang Xia, Yalong Guo, Chen Wei, Jianwei Cui, and Bin Wang. 2021. Towards an online empathetic chatbot with emotion causes. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2041–2045.

Zhaojiang Lin, Peng Xu, Genta Indra Winata, Farhad Bin Siddique, Zihan Liu, Jamin Shin, and Pascale Fung. 2020. Caire: An end-to-end empathetic chatbot. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13622–13623.

K Liu and Rosalind W Picard. 2005. Embedded empathy in continuous, interactive health assessment. In *CHI Workshop on HCI Challenges in Health Assessment*, volume 1, page 3.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2018. Towards empathetic open-domain conversation models: A new benchmark and dataset. *arXiv preprint arXiv:1811.00207*.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric M Smith, et al. 2020. Recipes for building an open-domain chatbot. *arXiv preprint arXiv:2004.13637*.

Haoyu Song, Wei-Nan Zhang, Yiming Cui, Dong Wang, and Ting Liu. 2019. Exploiting persona information for diverse generation of conversational responses. *arXiv preprint arXiv:1905.12188*.

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.

Weixuan Wang, Xiaoling Cai, Chong Hsuan Huang, Haoran Wang, Haonan Lu, Ximing Liu, and Wei Peng. 2021. Emily: Developing an emotion-affective open-domain chatbot with knowledge graph-based persona. *arXiv preprint arXiv:2109.08875*.

Peixiang Zhong, Chen Zhang, Hao Wang, Yong Liu, and Chunyan Miao. 2020. Towards persona-based empathetic conversational models. *arXiv preprint arXiv:2004.12316*.

Li Zhou, Jianfeng Gao, Di Li, and Heung-Yeung Shum. 2020. The design and implementation of xiaoice, an empathetic social chatbot. *Computational Linguistics*, 46(1):53–93.

An End-to-End Speech Recognition for the Nepali Language

Sunil Regmi[1] **Bal Krishna Bal**[2]

Information and Language Processing Research Lab
Department of Computer Science and Engineering
Kathmandu University
Dhulikhel, Kavre, Nepal
[1]sunilregmi233@gmail.com
[2]bal@ku.edu.np

Abstract

Most Nepali speech recognition systems have followed the traditional methods of Speech recognition which involve separately trained acoustic, pronunciation, and language model components. Developing such components from scratch requires domain expertise and is also time-consuming. Similarly, adoption of attention-based approaches, which is the latest technology trend is also not that popular in Nepali speech recognition. The only method found to be applied is the CTC method. In this work, we present an End-to-End ASR approach, which uses a joint CTC- attention-based encoder-decoder and a Recurrent Neural Network based language modeling through which we not only eliminate the need of creating a pronunciation lexicon from scratch but also take the fullest advantage of the state-of-the-art deep learning technologies. We use the ESPnet toolkit which uses Kaldi Style of data preparation framework. The speech and transcription data used for this research is freely available on the Open Speech and Language Resources (OpenSLR) website. We have obtained a Character Error Rate (CER) of 10.3% on 159k transcribed speech (159k utterances taken from OpenSLR.

1 Introduction

There has been increasing use of Automatic Speech Recognition (ASR) technologies in many application domains like Hospital Information Systems to IoT devices, industrial robotics, forensic, defense and aviation, etc. to name a few. According to Jelinek (1976), a conventional speech recognition system consists of several modules that comprise the acoustic, lexical, and language models supported by a probabilistic model for noisy channels. To build an acoustic model, we first need to build a Hidden Markov Model (HMM) and a Gaussian Mixture Model (GMM). In addition, the system requires linguistic knowledge based on a lexical model, usually based on a handmade pronunciation dictionary that does not have an explicit word limit. The lexical model requires language-specific tokenization modules for language modeling to develop ASR for new languages. Finally, decoding must be done with the synergistic action of all the modules, resulting in a complex decoding process (Hori, et al., 2017). Nevertheless, today's systems rely heavily on the End-to-End architectures that have emerged around traditional techniques(Geofferey et al., 2012). Unlike in the traditional methods like Hidden Markov Model (HMM) based model, the end-to-end approach addresses a single network architecture within a deep learning framework that directly maps language features to words or characters (Hori et al., 2017). There are two main architectures for end-to-end ASR. Connectionist Temporal Classification (CTC) allows the training of acoustic models without frame-level alignment between transcripts and acoustic frames, while attention models perform alignment between acoustic frames and identifiers (Wang & Li, 2019). Kim et al (2017) present a CTC/Attention-based collaborative end-to-end ASR that uses the CTC objective from loss function during the attention model training. The joint CTC-Attention-based encoder-decoder utilizes both the benefits of CTC and attention during training. The CTC predictions are also used in the decoding process. CTC can be interpreted as just a loss function used for training neural networks such as cross-entropy models. It is used in a difficult situation where the availability of aligned data is an issue, like in ASR. The capability to model temporal correlations with appropriate context information can be found in Convolutional Neural Network (CNN) (Ying et al., 2016; Zhang et al., 2017). Also, the language model (LM) integration is widely used for the HMM-based systems, something that is still applicable and effective for End-to-End ASR as well (Mikolov et al., 2010). The attention-based approach is used with CTC and Recurrent Neural Network-based Language Model (RNN-LM) as a joint decoder and a CNN-based shared encoder to achieve a state-of-the-art accuracy (Kim et al., 2017).

Most of the works on Speech Recognition for Nepali are based on the traditional methods. Nepali is written in the Devanagari script, which is essentially phonetic, so its pronunciation is very similar to how it is written. In Nepali, there are a total of 11 vowels and 33 consonants (Bal, 2004).

We use an End-to-End Speech Recognition Architecture that is based on a joint CTC/Attention-based encoder-decoder with a Recurrent Neural Network-based language modeling developed by Hori et al. (2017). We use the ESPnet framework (Watanabe et al., 2018; Hayashi et al., 2020; Inaguma, S., 2020; Chenda et al., 2021), which is an End-to-End speech processing toolkit. It sits on top of the Kaldi speech recognition toolkit (Povey et al., 2011) and the deep learning frameworks based on PyTorch (Paszke et al., 2019).

This paper is divided into five sections. Section I gives an introduction about the problem, section II gives an overview of the related works in Nepali speech recognition, section III describes the methodology for developing RNN-LM and CTC/attention-based models, section IV discusses the model and presents the experimental results, and finally section V presents the conclusion and future works.

2 Related Works

Some prior works on Nepali ASR in the character, word, and sentence level have been conducted. Google, for example, provides cloud-based Speech Recognition for more than 80 languages including Nepali. Unfortunately, there is not any publicly available documentation on the underlying methods and techniques used for the Project. Prajapati et al (2008) analyzed existing models for speech recognition and upon finding the shortcomings of Dynamic Time Wrapping (DTW's) approach, they proposed a new model called the Ear Model. They report to have obtained better accuracy than existing methods, but only for single alphabets. The classical, most commonly used model for Speech Recognition is Hidden Markov's model which is used by Ssarma et al (2017) where the authors have obtained a fairly good accuracy of 75% for isolated words. Similarly,, Regmi et al. (2019) used the RNN-CTC model and obtained a CER of nearly about 34% accuracy making use of the language model. Bhatta et al. (2020) proposed a model comprising CNN, GRU, and CTC networks. The dataset used by the authors is provided by Open Speech and Language Resources. Their build model recognizes speech with the WER of 11%. Baral & Shrestha (2020) present a comparative study of popular speech recognition methods for the Nepali language where they built a phonetic dictionary from scratch and presented the findings on 50K vocabulary for DNN and GMM based techniques with speaker adaptation. The experiments were carried out in the Kaldi toolkit. The lowest WER for different GMM- HMM models were 29.45% and similarly, the lowest WER for different DNN models using DNN- TDNN-LSTM was 11.55%.

Our study reveals that the researches till now, in Nepali speech recognition, have used the traditional methods except for Bhatta et al (2020) and Regmi et al (2019) who have used the CTC based End-to-End approach for small vocabulary tasks. The main difference between the CTC and attention method is that the conditional independence assumptions are not employed in the attention-based methods whereas CTC requires several conditional independence assumptions to get the probability of a label sequence. From this perspective, the attention mechanism, on the other hand, is surprisingly flexible as it allows extremely non-sequential alignments. However, for speech recognition tasks, the alignment is usually monotonic. As a regularization method, Kim et al (2017) uses a CTC objective from loss function to an attention-based encoder network which encourages the alignments' monotonicity. This method improves the accuracy of ASR compared to CTC or attention-based methods alone.

3 Architecture for Joint CTC/Attention Model

There are a series of steps involved in a speech recognition system requiring different components like data collection and preparation, data pre-processing, feature extraction, model building, training and testing, and decoding. For a Machine Learning System, clean and processed data are a basic prerequisite and thus represents a very crucial resource.

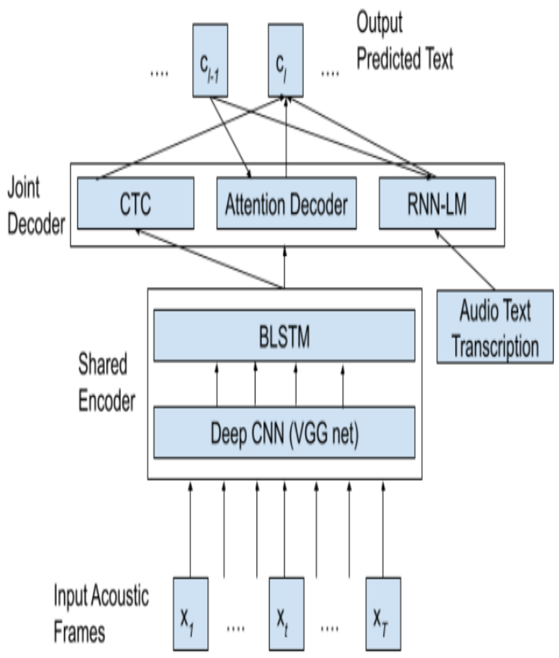


Figure 1: Joint CTC-attention ASR Architecture with Deep CNN and RNN-LM. Source: (Hori et al., 2017)

Audio data should basically be collected and recorded in a controlled environment, without background noise such as coughing, wheezing, and throat cleansing, or environmental sounds such as beeps, phone rings, and door slams. In the Nepali Language, there are some speech corpora provided by the Open Speech and Language Resource for research purposes. Fortunately, the speech data that we use from the aforementioned source is noise-free and thus ideal for our research.

In this research, a CTC-attention based joint encoder-decoder is used that takes advantage of both CTC and attention during training. For the language modeling part, Recurrent Neural Network is used. The RNN-LM network shows good improvement over the hybrid/HMM model and is merged in parallel with the attention decoder, which can be trained individually or jointly. The training is done with character sequences without word-level knowledge. Figure 1 shows the architecture of a CTC-attention based joint encoder-decoder (Hori et al., 2017). This approach's performance is superior to the different state-of-the-art hybrid ASR systems.

In the given architecture, at first analog electrical signals are converted to digital signals. This is done in the feature extraction part where Mel Frequency Cepstral Coefficients (MFCC) are used to extract audio features to distinguish different sounds or letters of a language. After that, the features are passed to a Deep CNN-based encoder that uses the Visual Geometry Group (VGG) network. The input for the CNN network is the Mel-scale feature from the raw speech features. The initial layers of the VGG net architecture (K. Simonyan and A. Zisserman, 2014) with 6-layer CNN architecture are used followed by BLSTM layers in the encoder network (Hori et al., 2017). The output is used by the CTC and the Attention Decoder as a shared encoder. The joint CTC-Attention-based encoder-decoder utilizes both the benefits of CTC and attention during training. The CTC predictions are also used in the decoding process. The model outputs the L-length character sequence as a set of individual characters U.

U = {'अ', 'आ', 'इ', 'ई', 'उ', 'ऊ', 'ए', 'ऐ', 'ओ', 'औ', 'क', 'ख', 'ग', 'घ', 'ङ', 'च', 'छ', 'ज', 'झ', 'ञ', 'ट', 'ठ', 'ड', 'ढ', 'ण', 'त', 'थ', 'द', 'ध', 'न ', 'प', 'फ', 'ब', 'भ', 'म', 'य', 'र', 'ल', 'व', 'श', 'ष', 'स', 'ह', '◌ँ', '◌ं ', '◌ः', '◌़', '◌ा', '◌ि', '◌ी', '◌ू', '◌ृ', '◌े', '◌ै', '◌ो', '◌ौ', '◌्', 'ॐ', 'ऋ', '।', '०', '१', '२', '३', '४', '५', '६', '७', '८', '९', ' '}

Out of 129 Unicode characters, 71 characters are used which are extracted from the text transcription of the speech data used in this research. The character set is indexed from 0 to 70. The input of CTC is the last hidden layer of the Bidirectional Long-Short Term Memory (BLSTM). Joint decoding is performed with a one-pass beam search algorithm that combines both attention-based scores and CTC scores to further eliminate irregular alignments. Use one tuning parameter to linearly interpolate both objectives, the multitasking learning (MTL) rate, which is typically set to 0.3. For the language modeling part, Recurrent Neural Network is used. The RNN-LM probabilities of output label prediction are used in conjunction with the decoder network because they assign a probability to each clause in such a way that the more probable strings (in a sense) get a higher probability and we tend to choose one. Similarly, an additional rescoring step is not needed if we combine the LM probabilities while decoding (Hori et al., 2017). Thus, this model can be viewed as a single gigantic neural network, even though its parts are pretrained independently.

4 Experiments and Results

The audio data and the text transcription are collected from the openslr.org website which contains transcribed audio data for Nepali Language Kjartansson et al., Sodimana et al (2018). The dataset contains Nepali Speech Data containing ~157K utterances and a text file that contains the utterance id and the text of the respective speech utterances. All speech utterance sums up to speech of duration of 9,278 minutes and 11 seconds. This corpus contains 86,062 unique utterances. Nepali Speech corpus from Open speech and language resource named "Multi-speaker TTS data for Nepali (ne-NP)" (SLR43) has been also used to the train models. It contains about 2064 long sentences-based utterance spoken by 18 different female speakers. All speech utterances sum up to speech of duration of 167 minutes and 45 seconds. This corpus contains 2064 unique utterances. Altogether, ~159k utterances are used.

For conducting the experiments, RTX 2060 GPU is used for training the model and Ryzen 9 CPU with 16 cores is used for the decoding process. The dataset was split in the ratio of 8:2 for the train and test dataset and from the remaining 80% of the training data, again it was split for train and validation set in a ratio of 8:2. The CNN BLSTM encoder uses 80 mel-scale filter banks with the delta and delta-delta features as input features. A 4-layer BLSTM with 320 units per layer and direction is used. To extract the convolutional features, 10 centered convolutional filters with a width of 100 were used. A 1-layer LSTM with 320 cells units are used as a decoder network. A single-layer LSTMs for RNN-LMs are individually trained using transcription, combined with a decoder network, and optionally retrained in collaboration with encoders, CTC networks, and decoders. The training is done for 20 epochs with the patience of value 2. Here, no extra text data were used but the use of additional un-transcribed data can further improve the results. The AdaDelta algorithm with gradient clipping was used for the optimization (Hori et al., 2017). The beam width and CTC-weight were set to 20 and 0.3 in decoding process. The CER is used as the evaluation metric which is defined as the sum of characters that is substituted, inserted, and deleted in particular sentence divided by the entire number of characters within the dataset.

	Sub	Del	Ins	Total	Total/Total Characters	% CER
Validation	28317	16652	8757	53726	53726/458271	11.7
Test	31117	18627	9739	59483	59483/576870	10.3

Table 1: CER using joint CTC/Attention model for mixed SLR54 and SLR43 datasets (~159K utterances)

Ground Truth	Prediction
कानपुर भारतका सर्वाधिक	कानपुर भारतका सर्वाधिक
पत्ता लगाउनको लागि	पत्ता लगाउनको लागि
२ हजार ३ गरी	२००० ३ गरी
छोराहरूको भविष्य भनी	चोराहरूको भविष्य बने
सूर्य चन्द्रमा र पृथ्वीका माझ उत्पन्न हुने दुईवटा छाँयालाई राहु र केतु भनिन्छ।	सूर्य चन्द्रमा र पृथ्वीका माझ उत्पन्न हुने देवटा छयालाई राहु र केतु भनिन्छ।
खसी बाख्रालाई घाँस	खसी बाख्रालाई गाँस

Table 2: Ground truth and prediction for sample test data

Table 1 provides the CER for the mixed SLR54 and SLR43 datasets. This model recognizes Nepali speech input data with 10.3% CER. Similarly, Table 2 provides a sample of ground truth and prediction for the sample test data. There are cases where the joint CTC/Attention model with 10.3 % CER in test data incorrectly predicts the utterance - २ हजार ३ गरी as २००० ३ गरी. Here, the 2 हजार is predicted as २००० which is not necessarily wrong but this can affect the accuracy of the model. We also noted that the model sometimes gets confused with छ and recognizes as च, भ as ब, and घ as ग. In some cases, the characters ँ is not recognized by the model. The obtained CER can be further improved by fine-tuning the parameters i.e, ctc-weight and multi-task learning rate.

5 Conclusion and Future Works

We experimented with the Nepali speech datasets using the End-to-End Automatic Speech Recognition framework called ESPnet. The framework uses CTC, attention, their joint form with RNN based LM, transformer, conformer, and transducer-based models. In addition, we experimented with an advanced architecture that includes common decoding, a deep CNN encoder, and an RNN-LM network proposed by Hori et al (2017). The proposed approach eradicates the need of components which are essential in any conventional ASR model. We have achieved a speech recognition of CER – 11.7% and 10.3% for Nepali Language, respectively for the validation and the test data using the End-to-End model. Also, using substantial amounts of unlabeled data in conjunction with a pre-trained RNN-LM, this model can be improved further.

We recommend future works on End-to-End speech recognition for Nepali to be focused on employing the transformer, conformer, and transducer-based models. In recent studies, the conformer (Convolution-augmented Transformer) (Gulati et al., 2020) network showed a significant improvement and has outperformed the performance of Transformer and CNN-based models with different ASR standard datasets (Guo et al., 2021). It is also recommended to use multiple GPUs for fast training and decoding time. Additionally, the toolkit "ESPRESSO" suggested by Yiming et al. (2019) can be used to gain 4x faster accuracy in decoding instead of the ESPnet.

References

Bal, Bal Krishna. (2004). *Structure of Nepali Grammar.* PAN Localization, Working Papers 2004-2007, 332– 396.

Baral, Elina & Shrestha, Sagar (2020). Large Vocabulary Continuous Speech Recognition for Nepali Language. 8(4), 68–73. https://doi.org/10.18178/ijsps.8.4.68-73

Bhatta, R. (2020). Nepali Speech Recognition Using CNN, GRU and CTC. In Proceedings of the 32nd Conference on Computational Linguistics and Speech Processing (ROCLING 2020) (pp. 238–246). The Association for Computational Linguistics and Chinese Language Processing (ACLCLP).

Chenda Li, Jing Shi, Wangyou Zhang, Aswin Shanmugam Subramanian, Xuankai Chang, Naoyuki Kamo, Moto Hira, Tomoki Hayashi, Christoph Boeddeker, Zhuo Chen, & Shinji Watanabe (2021). ESPnet-SE: End-to-End Speech Enhancement and Separation Toolkit Designed for ASR Integration. In Proceedings of IEEE Spoken Language Technology Workshop (SLT) (pp. 785–792).

Frederick Jelinek. 1976. Continuous speech recognition by statistical methods. Proceedings of the IEEE 64(4):532–556

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine 29(6):82–97.

Gulati, A., Qin, J., Chiu, C. C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., & Pang, R. (2020). *Conformer: Convolution-augmented transformer for speech recognition.* Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, 2020-Octob, 5036–5040. https://doi.org/10.21437/Interspeech.2020-3015

Guo, P., Boyer, F., Chang, X., Hayashi, T., Higuchi, Y., Inaguma, H., Kamo, N., Li, C., Garcia-Romero, D., Shi, J., Shi, J., Watanabe, S., Wei, K., Zhang, W., & Zhang, Y. (2021). *Recent Developments on Espnet Toolkit Boosted By Conformer*. 5874–5878. https://doi.org/10.1109/icassp39728.2021.9414858

Hayashi, T., Yamamoto, R., Inoue, K., Yoshimura, T., Watanabe, S., Toda, T., Takeda, K., Zhang, Y., & Tan, X. (2020). Espnet-TTS: Unified, reproducible, and integratable open source end-to-end text-to-speech toolkit. In Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 7654–7658).

Hori, T., Cho, J., & Watanabe, S. (2019). *End-to-End Speech Recognition with Word-Based Rnn Language Models.* 2018 IEEE Spoken Language Technology Workshop, SLT 2018 - Proceedings, 389–396. https://doi.org/10.1109/SLT.2018.8639693

Hori, T., Watanabe, S., & Hershey, J. R. (2017). *Joint CTC/attention decoding for End-to-End speech recognition.* ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers), 1, 518–529. https://doi.org/10.18653/v1/P17-1048

Hori, T., Watanabe, S., Zhang, Y., & Chan, W. (2017). *Advances in joint CTC-attention based end-to-end speech recognition with a deep CNN encoder and RNN-LM.* Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, 2017- Augus, 949–953. https://doi.org/10.21437/Interspeech.2017-1296

Inaguma, S. (2020). ESPnet-ST: All-in-One Speech Translation Toolkit. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations (pp. 302–311). Association for Computational Linguistics.

Kim, S., Hori, T., & Watanabe, S. (2017). Joint CTC-attention based End-to-End speech recognition using multi-task learning. ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, 4835–4839. https://doi.org/10.1109/ICASSP.2017.7953075

K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

Kjartansson, O., Sarin, S., Pipatsrisawat, K., Jansche, M., & Ha, L. (2018). Crowd-Sourced Speech Corpora for Javanese, Sundanese, Sinhala, Nepali, and Bangladeshi Bengali. August, 52–55. https://doi.org/10.21437/sltu.2018-11

Mikolov, Tomas & Karafiát, Martin & Burget, Lukas & Cernocký, Jan & Khudanpur, Sanjeev. (2010). Recurrent neural network based language model. Proceedings of the 11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010. 2. 1045-1048.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). *PyTorch: An imperative style, high-performance deep learning library.* ArXiv, NeurIPS.

Povey, D., Boulianne, G., Burget, L., Motlicek, P., & Schwarz, P. (2011). The Kaldi Speech Recognition. IEEE 2011 Workshop on Automatic Speech Recognition and Understanding, January. http://kaldi.sf.net/

Prajapati, C., Nyoupane, J., Shrestha, J. D., & Jha, S. (2008). Acknowledgment. 24208.

Regmi, P., Dahal, A., & Joshi, B. (2019). Nepali Speech Recognition using RNN-CTC Model. International Journal of Computer Applications, 178(31), 1–6. https://doi.org/10.5120/ijca2019918401

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, 1–14.

Sodimana, K., De Silva, P., Sarin, S., Kjartansson, O., Jansche, M., Pipatsrisawat, K., & Ha, L. (2018). A Step-by-Step Process for Building TTS Voices Using Open Source Data and Frameworks for Bangla, Javanese, Khmer, Nepali, Sinhala, and Sundanese. 66–70. https://doi.org/10.21437/sltu.2018-14

Ssarma, M. K., Gajurel, A., Pokhrel, A., & Joshi, B. (2017). HMM based isolated word Nepali speech recognition. Proceedings of 2017 International Conference on Machine Learning and Cybernetics, ICMLC 2017, 1, 71–76. https://doi.org/10.1109/ICMLC.2017.8107745

Wang, S., & Li, G. (2019). Overview of End-to-End speech recognition. Journal of Physics: Conference Series, 1187(5). https://doi.org/10.1088/1742-6596/1187/5/052068

Watanabe, S., Hori, T., Karita, S., Hayashi, T., Nishitoba, J., Unno, Y., Soplin, N. E. Y., Heymann, J., Wiesner, M., Chen, N., Renduchintala, A., & Ochiai, T. (2018). *ESPNet: End-to-End speech processing toolkit.* Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, 2018-Septe,2207–2211. https://doi.org/10.21437/Interspeech.2018-1456

Yiming Wang, Tongfei Chen, Hainan Xu, Shuoyang Ding, Hang Lv, Yiwen Shao, Nanyun Peng, Lei Xie, Shinji Watanabe, & Sanjeev Khudanpur (2019). Espresso: A Fast End-to-end Neural Speech Recognition Toolkit. In 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU).

Zhang, Ying, Pezeshki, M., Brakel, P., Zhang, S., Laurent, C., Bengio, Y., & Courville, A. (2016). *Towards End-to-End speech recognition with deep convolutional neural networks*. Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, 08-12-Sept,410–414. https://doi.org/10.21437/Interspeech.2016-1446

Zhang, Yu, Chan, W., & Jaitly, N. (2017). *Very deep convolutional networks for End-to-End speech recognition.* ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, 4845–4849. https://doi.org/10.1109/ICASSP.2017.7953077

Impact of Microphone position Measurement Error on Multi Channel Distant Speech Recognition & Intelligibility

Karan Nathwani
IIT Jammu,
India
karan.nathwani@iitjammu.ac.in

Sunil Kumar Kopparapu
TCS Research,
Tata Consultancy Services Ltd., India.
sunilkumar.kopparapu@tcs.com

Abstract

It was shown in (Raikar et al., 2020) that the measurement error in the microphone position affected the room impulse response (RIR) which in turn affected the single channel speech recognition. In this paper, we extend this to study the more complex and realistic scenario of multi channel distant speech recognition. Specifically we simulate m speakers in a given room with n microphones speaking without overlap. The n channel audio is beamformed and passed through a speech to text (s2t) engine. We compare the s2t accuracy when the microphone locations are known exactly (ground truth) with the s2t accuracy when there is a measurement error in the location of the microphone. We report the performance of an end-to-end s2t on beamformed input in terms of character error rate (CER) and and also speech intelligibility and quality in terms of STOI and PESQ respectively.

1 Introduction

The multi-path reflections (attributed by RT60) in an open enclosure is caused during hands free speech communication. Such multi-path reflections along side noise impinging on multiple microphones result in noisy reverberated speech which has a deteriorating impact on the distant speech recognition performance (Naylor and Gaubitch, 2010). Further, the quality and intelligibility of the speech in hands free communication would also deteriorate (Nathwani et al., 2017, 2016; Biswas et al., 2021). There is an urgent need for noise reduction, dereverberation and conjunction of both during communications. In this context, multi-microphone-based approaches exploiting spatial acoustic cues such as spatial diversity, inter-intensity differences and inter-time differences, receives particular interest

Accurate estimation of RT60 plays an important role in several applications like (a) sound reproduction of geometry aware room (Betlehem and Abhayapala, 2005; Tang et al., 2020; Kim et al., 2019), (b) reconstruction of the room geometry (Crocco et al., 2014; Moore et al., 2013; Yu and Kleijn, 2019), (c) robust automatic speech recognition (ASR) (Yoshioka et al., 2012; Krueger and Haeb-Umbach, 2010; Heymann et al., 2019) and (d) speech enhancement (Zhang et al., 2017; Li and Koishida, 2020; Gannot et al., 2017). In a single channel (microphone) scenario, several techniques exist to estimate the room impulse response (RIR) (Szöke et al., 2019) when the microphone position is not erroneous. Though, given the room geometry, RIR computing is non-trivial; RIR estimates require the exact location of both the source and the microphone. A comparative study for blind reverberation time estimation in single microphone scenario is explored in (Löllmann et al., 2019). A slight displacement (due to human interventions or due to routine maintenance etc. (Raikar et al., 2020; Muthukumarasamy and Donohue, 2009; Sachar et al., 2002)) in the microphone position could severely hamper the RIR estimates (Muthukumarasamy and Donohue, 2009; Sachar et al., 2002). In particular, the impact of measurement error in microphone position on speech intelligibility and quality is explored in (Raikar et al., 2020).

In practical applications, a microphone array in comparison to single microphone, grants more benefits (Nathwani et al., 2013; Stoica et al., 2002) especially due to the spatial information and associated applications like directional or arrival (DOA), location of sound source and room information (Pavlidi et al., 2013; Chen et al., 2015). However, as in single microphone case, calibration error because of displaced microphone array is not trivial to model in a real time scenarios (Sachar et al., 2004, 2002). This is, primarily because it is computationally expensive and prone to error. In (Muthukumarasamy and Donohue, 2009), a delay and sum beamforming (DSB) technique is used to model location errors analytically, they show that the measurement error in microphone position affects the intelligibility and quality due to change in the overall RIR. DSB approach has two drawbacks, namely, it requires (a) a large number of sensors to improve the SNR and (b) it cannot adapt to varying noisy conditions.

To overcome the limitations of DSB, adaptive beamformers like capon (Stoica et al., 2002) and minimum-variance distortionless response (MVDR) beamformers have been introduced to perform joint noise and reverberation cancellation. In (Schwartz et al., 2014), a joint noise cancellation and dereverberation is illustrated in generalized side lobe canceller (GSC) framework, while in (Schwartz et al., 2015), a nested structure in the GSC framework is proposed. As opposed to beamforming, there are adaptive filtering based approaches that do not require spatial information of the speech source. In (Dietzen et al., 2017), a multi-channel linear prediction (MCLP) in Kalman Filtering domain is proposed for blind dereverberation. However, they fail to perform in the presence of noise as they focus only on reverberation.

Multi-channel beamformers are prone to measurement error due to change in microphone array position, which affects the RIR. This brings to focus, the question, *does microphone measurement error affect beamforming performance?* In particular, the impact of such displacement error, for single microphone channel, on speech intelligibility and quality has been explored in (Raikar et al., 2020). As an extension, it is of interest to explore and investigate the performance of DSB and adaptive beamformer (MVDR) for multi channel distant automatic speech recognition (ASR), intelligibility and quality. Towards this study, we simulate m speakers in a given room with n microphones speaking without overlap. The output of n channel audio is beamformed and passed through a speech to text (s2t) engine.

We compare the s2t accuracy when the microphone locations are known exactly (i.e. ground truth) with the s2t accuracy, when there is a measurement error in the location of the microphone location. The experimental results illustrate that the measurement error in microphone position has a significant effect on s2t performance. Consequently, the main contribution of this paper is the formulation of the problem to enable analysis of the effect of microphone position measurement error on distant speech recognition as well as speech intelligibility and quality. Note that in this paper, we make no attempt to introduce a new technique or algorithm to improve the distant speech recognition and intelligibility scores; rather the experimental studies reported in this paper should allow for development of new techniques in the future.

2 Problem Formulation

Let us assume a room $\mathcal{R}(L, W, H)$ of dimension $L \times W \times H$. Let there be N, $s_1, s_2, \cdots s_N$ speakers located at $\{(x_i^s, y_i^s, z_i^s)\}_{i=1}^N$ respectively and M microphones, $r_1, r_2, \cdots r_M$, located at $\{(x_j^r, y_j^r, z_j^r)\}_{j=1}^M$ respectively in the room $\mathcal{R}$. Let $u_i(t)$ be the utterance spoken by the speaker s_i at location (x_i^s, y_i^s, z_i^s) and let h_{kl} be the RIR computed for the speaker s_k and microphone r_l pair. Let o_l be the speech recorded at the microphone r_l. We can now write the output at the M microphones as $[o_1(t), o_2(t), \cdots, o_M(t)]^T =$

$$\begin{bmatrix} h_{11} & h_{21} & \cdots & h_{N1} \\ h_{12} & h_{22} & \cdots & h_{N2} \\ \vdots & \vdots & \ddots & \vdots \\ h_{M1} & h_{M2} & \cdots & h_{NM} \end{bmatrix} * \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_N(t) \end{bmatrix}$$

where $*$ is the convolution operator such that

$$o_l(t) = \sum_{i=1}^{N} h_{il} * u_i(t). \tag{1}$$

Note that h_{kl} is the RIR and is a function of c the speed of sound, f_s the sampling frequency of utterance, $L \times W \times H$ volume of the room, β the reverberation time, (x^s, y^s, z^s) the location of the speaker, and (x^r, y^r, z^r) the rectangular coordinates of the microphone. RIR $h_{kl} =$

$$\texttt{rir\_gen}(c, f_s, (x_l^r, y_l^r, z_l^r), (x_k^s, y_k^s, z_k^s), L, \beta) \tag{2}$$

Standard utilities to simulate h are readily available (Habets, 2006) and as mentioned in (Raikar et al., 2020) h_{kl} is prone to measurement errors in the position of the microphone, namely (x^r, y^r, z^r) as seen in (2).

Let an error $\epsilon = [\epsilon_x, \epsilon_y, \epsilon_z]$ be made in measuring the position of the l^{th} microphone r_l, so the measured location of the r_l is $r_{l\epsilon} = [x_l^r + \epsilon_x, y_l^r + \epsilon_y, z_l^r + \epsilon_z]$. Subsequently there is an error introduced in the RIR, namely, $h_{*l\epsilon} =$

`rir_gen`$(c, f_s, r_{l\epsilon}, s_*, L, \beta, n)$. Clearly an error in measurement of the microphone $r_{l\epsilon}$ results in an error in the output speech, namely,

$$o_{l\epsilon}(t) = \sum_{i=1}^{N} h_{il\epsilon} * u_i(t). \tag{3}$$

The actual output of the l^{th} microphone, if there were no measurement errors, is (1). Also in all our experiments we assume ϵ to be Gaussian $\mathcal{N}(0, \sigma^2)$ with 0 mean and $\sigma^2 = 0.1, 0.5, 1$ as was done in (Raikar et al., 2020).

Multi channel distant speech recognition involves beamforming ($\mathcal{B}$) the multi channel speech, namely, $(o_1, o_2, \cdots, o_M)$ from M microphones, to form an equivalent of a close microphone (single channel) speech followed by ASR (`s2t`). For convenience let us represent this process as

$$\mathcal{T} = \texttt{s2t}(\mathcal{B}(o_1, o_2, \cdots o_l \cdots o_M)) \tag{4}$$

As can be seen an error $(\epsilon_x, \epsilon_y, \epsilon_z)$ in measuring the position of a microphone results in an error at the output of the microphone, namely, $o_{l\epsilon}$ (3), this results in an error in speech recognition output T_ϵ, namely

$$T_\epsilon = \texttt{s2t}(\mathcal{B}(o_{1\epsilon}, o_{2\epsilon}, \cdots o_{l\epsilon} \cdots o_{M\epsilon})) \tag{5}$$

In this paper, we analyze the error in the recognition (5) of speech because of an error in measurement of the location of the microphone.

3 Experimental Results and Discussion

3.1 Experimental Setup

We assume[1] a room of dimension $5 \times 5 \times 5\ m^3$ and $M = 4$ microphones and $N = 2$ speakers. Unlike in a microphone array setup we assume that the microphones can be located anywhere in the room, preferably closer to the walls and the speakers are inside the room. As an example, the room and the location of microphone and the speakers is show in Figure 1 (a) corresponding to the location of microphones and speakers shown in Table 1.

Let $u_i(t)$ be the utterance spoken speaker s_i and define $\lambda(t)$ (Figure 1 (b)) to be an arbitrary multi-valued function (the number of values depend on the number of speakers, in our case 2 corresponding to the two speakers). As seen in Figure 1 (b) s_1 (s_2) is active during the time interval when $\lambda(t) = 1 (\lambda(t) = 2)$ where $\lambda_i(t)$ is

$$\begin{aligned} &= 1 \quad \text{for} \quad \lambda(t) = i \\ &= 0 \quad \text{for} \quad \lambda(t) \neq i \end{aligned} \tag{6}$$

Let $\bar{u}_i(t) = u_i(t)\lambda_i(t)$ represents the utterance of speaker s_i (the duration for which speaker s_i was active). In all our experiments we construct the speech utterance as

$$U(t) = \sum_i \bar{u}_i(t) \tag{7}$$

Subsequently, we construct the multi-channel output (4 channels) as

$$o_j = \sum_i \bar{u}_i(t) * h_{ij} \quad \text{for} \quad \forall j = 1, 2, 3, 4 \tag{8}$$

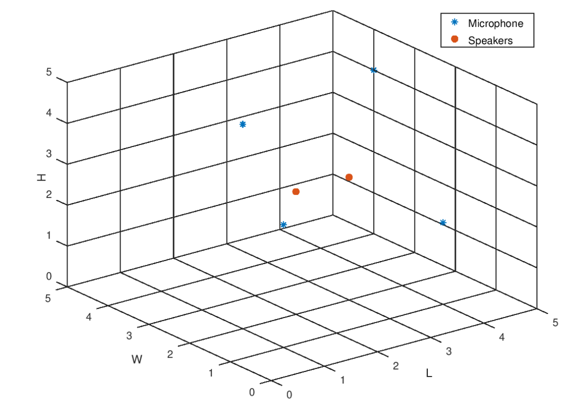
Let $\mathcal{T}_g = \texttt{s2t}(U(t))$ be the transcription of the utterance $U(t)$ which we consider as the ground truth. Now we get $\mathcal{T} = \texttt{s2t}(\mathcal{B}(o_1, o_2, o_3, o_4))$ when there is no error in the measurements of the location of the microphones. And $\mathcal{T}_\epsilon = \texttt{s2t}(\mathcal{B}(o_{1\epsilon}, o_{2\epsilon}, o_{3\epsilon}, o_{4\epsilon}))$ when there is an error (ϵ) in measurements of the location of the microphones as mentioned in Section 2. We experiment with two different beamformers, namely, (a) delay and sum (DSB) and (b) minimum variance distortionless response (MVDR) (Kumatani et al., 2015; Wei et al., 2021) (namely, $\mathcal{B} \in \{\text{DSB}, \text{MVDR}\}$). The MVDR is an adaptive beamformer which optimizes the desired speech in a given direction by filtering out interfering signal (Wei et al., 2021). This is achieved by selecting the weights of beamformer with the idea of minimizing the output power under the constraint that the target speech is unaffected. On the other hand, DSB is a fixed beamformer which is quite effective when the environment only contains uncorrelated noise between microphones (Wei et al., 2021) which is the case in our study.

We use an end-to-end transformer based state-of-the-art speech to alphabet engine for `s2t` (Hugging Face Team). The `s2t` inference is based on the greedy Connectionist temporal classification

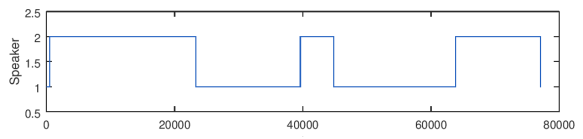
Table 1: Microphone and Speaker location used in our Experiments (Figure 1(a)).

Microphone/Speaker	Location
r_1, r_2	(1, 2, 5), (5, 4, 4)
r_3, r_4	(4, 1, 2), (1, 1, 3)
s_1, s_2	(2, 2, 3), (3, 2, 3)

[1] though not realistic, it is common, in literature to assume a cuboid room dimension



(a)



(b)

Figure 1: Experimental Setup. (a) Four microphones and two speakers (Table 1) in a room of dimension $5 \times 5 \times 5$ (we assume a cuboid so that we are consistent with (Raikar et al., 2020)), (b) A sample $\lambda(t)$ where the x-axis represents the time (expressed in samples) and y-axis can take a value of 1 or 2 depending on who is speaking. Note that at any given time only one of the speaker is speaking (no overlap).

(CTC) output without the use of a language model. We compute the character error rate $\text{CER}(\mathcal{T}_g, \mathcal{T})$ (yi Wang et al., 2003) using the state of the art speech to alphabet engine (Hugging Face Team) when (a) there is no microphone location measurement error and $\text{CER}(\mathcal{T}_g, \mathcal{T}_\epsilon)$ and (b) when there is a measurement error in the location of the microphone. We hypothesize that the CER degrades with increased ($\epsilon \equiv \sigma^2$) measurement error in microphone location. We conducted a number of experiments using the above mentioned experimental setup. We first assumed the measurement error in microphone position has a Gaussian distribution with different variances (σ^2). We study the degradation of CER, the speech intelligibility, and speech quality as a function of the microphone location measurement error σ^2.

3.2 Data

We used the popular LibreSpeech database (OpenSLR, 2021) to generate real utterances as mentioned in the experimental setup. We randomly selected two audio files u_1, u_2 (when the duration is less than 5 s we append zeros to make them of duration 5 s) from the LibreSpeech clean dataset and constructed $U(t)$ of duration 5 s (see Algorithm 1). All experiments were conducted on 100 audio samples generated in this way and all results reported (Table 2 and 3) are averaged over these samples.

Algorithm 1: Constructing $U(t)$ (7).

```
input  : $L = 80000$ (5 seconds);
         $u_1(t), u_2(t)$
output : $U(t)$ of length 5 s
$ind = round\left(\frac{L}{6} * [1:6]\right)$;
$t_1 = randi([0, ind(1)])$;
for $ind \leftarrow 1$ to $length(ind)$ do
    $t_{i+1} = randi([ind(i), ind(i+1)])$;
end
$U(t) = [u_1([1:t_1]); u_2([t_1:t_2])]); u_1([t_2:$
  $t_3]); u_2([t_3:t_4]); u_1([t_4:t_5]); u_2([t_5:$
  $t_6]); u_1([t_6; L])]$;
```

3.3 Experimental Validation

We evaluate the distant speech recognition performance for the experimental setup (see Figure 1). The distant speech recognition performance are presented in the form of CER averaged over 100 audio samples (Table 2). Further, the validation of CER is achieved by computing the impact of microphone position measurement error on intelligibility and quality (Table 3).

To measure speech intelligibility, the well known (a) short time objective intelligibility (STOI) (Taghia and Martin, 2014) and (b) mutual information (MI) (Taghia et al., 2012) are used. STOI can take a value between 0 (completely unintelligible) and 1 (perfect intelligibility) and depends on the average amount of speech information available to a listener (Taal et al., 2010). The MI scores are estimated by first transforming the input signals into 15 sub-bands by using a $1/3$ octave band filter bank. Thereafter, the MI between the amplitude envelopes of the reference signal (7) and the beamformed signal with no microphone position error (bs, namely, $\mathcal{B}(o_1, o_2, o_3, o_4)$) and beamformed signal with microphone position error (bs_ϵ, namely, $\mathcal{B}(o_{1\epsilon}, o_{2\epsilon}, o_{3\epsilon}, o_{4\epsilon})$) are computed. MI is estimated per sub-band to evaluate the auditory perception (Kumatani et al., 2008). For speech quality assessment, we have used perceptual evaluation of speech quality (PESQ), signal to distortion ratio (SDR) and log-likelihood ratio (LLR). In PESQ, the speech signal is analyzed sample-by-sample after temporal alignment of corresponding excerpts of the original signal *w.r.t* to bs_ϵ and bs. In principal, PESQ models a mean opinion score (MOS) that ranges from 1 (bad) to 5 (excellent). Thereafter, we have used LLR objective measure which forms the distance measures. The LLR computes the spectral envelope difference between the original signal *w.r.t* bs_ϵ and bs (Gannot et al., 2001).

3.3.1 Speech Recognition Performance

The performance of distant speech recognition is computed in the form of CER for 100 random runs. It may be noted that lower values of CER suggest better performance. From Table 2, it can be seen that with higher measurement error (higher σ^2), the CER scores increase for both DSB and MVDR beamformers. We observe a maximum of 3% and 9% change in CER for MVDR and DSB respectively at $\sigma^2 = 1$ compared to when there is no measurement error in the microphone ($\sigma^2 = 0$). Comparing MVDR and DSB beamformers, it is observed that MVDR (adaptive beamformer) is not able to achieve the performance displayed by DSB. This lower performance can be attributed to the fact that MVDR is highly susceptible to singularity of the inverse matrix being used to calculate the weight matrix. This may result in musical noise or artifacts in the reconstruction. Figure 2 illustrates the box plot across 100 runs for DSB only (note that DSB outperforms MVDR in terms of CER). It can also be noticed that with increased σ^2 (0.5 or more), the variance and outliers in CER increase. Moreover at $\sigma^2 = 1$, the variations in CER is significantly high.

Table 2: Mean CER(%) for DSB and MVDR with varying microphone position measurement errors ($\sigma^2 = 0 \rightarrow$ no error).

$\sigma^2 \rightarrow$	0	0.01	0.5	1
DSB	26.23	27.12	27.58	28.92
MVDR	32.77	33.95	33.42	33.82

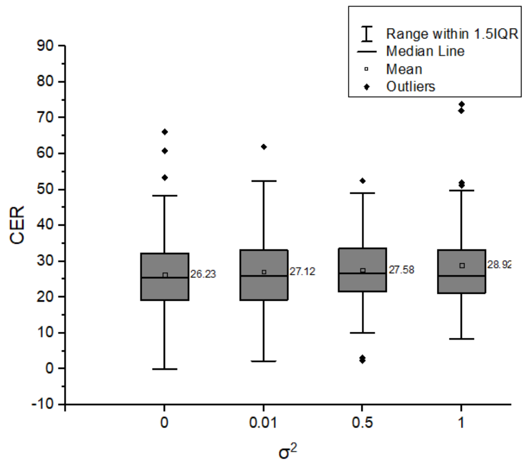


Figure 2: Variations of CER with the varying σ^2 for DSB.

Further to study how the microphone position measurement error compares with ambient or environmental noise effecting $U(t)$, we computed CER for noisy $U(t)$. We constructed $U_\epsilon(t) = U(t) + n(t)$, where $n(t)$ is an additive white Gaussian noise with different noise levels. We computed the CER on $\texttt{s2t}(U_\epsilon(t))$ with different noise levels (Figure 3). As expected, with an increase in the SNR levels (better signal strength), the CER scores decrease (better recognition) significantly. It can be also observed that with better signal (high SNRs), the outliers and variance in CER (computed over 100 runs; Figure 3) also decrease significantly, suggesting consistency in CER performance with increased signal strength. Also comparing the CER values in Table 2 and Figure 3, one can hypothesise that the measurement error in the microphone position is equivalent to an ambient noise of between 5 and 10 dB effecting the original signal. To further verify the impact of number of random runs (previously 100), we increased the random runs to 1000. It is observed that there is no difference in

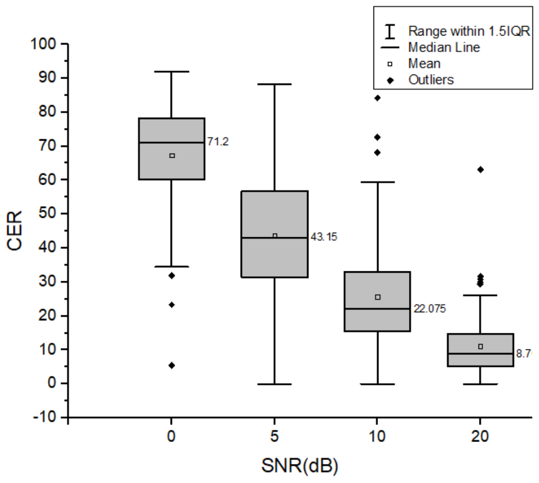


Figure 3: Variations of CER with the different noise levels.

CER between $\mathtt{s2t}(U_\epsilon(t))$ for increase number of runs.

It may also be emphasized from Table 2 that although MVDR being less sensitive to position/directional errors, their performance is still not satisfactory in comparison to DSB. One of the plausible argument is that the locations of microphones are fixed once the displacement or no displacement is made in microphone position. Hence it might be possible that the MVDR beamformer would not change the weights significantly once the locations of microphones are fixed. A more extensive analysis is deferred to future work to understand the underlying reasons for such a directional behaviour of MVDR in comparison to DSB.

3.3.2 Speech Intelligibility Performance

With an aim to answer the following question, namely, (a) Does the change in the microphone position impact the speech quality and speech intelligibility? (b) Is there any relationship between CER scores and speech quality (intelligibility) scores? and (c) How does the intelligibility and quality vary with respect to the two beamformers? To address these questions, we measured the speech intelligibility and speech quality with varying microphone measurement errors. Table 3 captures the mean (variance) scores of speech intelligibility and speech quality for varying microphone position measurement errors and for the two different beamformers (namely, DSB and MVDR). It may be noted that higher the STOI and MI scores, better is the intelligibility. On the other hand, higher the SDR, PESQ and lower the LLR scores, better is the quality of the speech signal.

From Table 3, it can be observed that DSB holds better speech quality while on the other hand MVDR claims better speech intelligibility. However with increasing microphone position measurement error (increasing σ^2), both MVDR and DSB performances for intelligibility and quality degrades significantly. In particular for STOI, the maximum degradation in DSB and MVDR performances is observed to 25% and 49% respectively, when we move from no microphone position error ($\sigma^2 = 0$) to $\sigma^2 = 1$. Similarly, this degradation in quality (SDR scores) for DSB and MVDR goes to 20% and 3% respectively.

Interestingly, it is observed that the quality, intelligibility and CER scores of both the beamformers do not change significantly, while error in microphone position varies from $\sigma^2 = 0.5$ to $\sigma^2 = 1$. These results indicate that the convergence in the error in the microphone position is achieved after $\sigma^2 = 0.5$. Further, we also able to verify the claim made in (Loizou and Kim, 2010) that non-correlation between improvement in quality and improvement in intelligibility. It is clearly visible from the MVDR and DSB performances in Table 3.

Similar to Figure 3, we also address how variation in microphone position error compares with the effect of environmental noise on intelligibility and quality. To achieve this, intelligibility and quality measures are computed between $U(t)$ and $U_\epsilon(t)$. It can be seen from Table 4 that as SNR increases, the mean intelligibility and quality scores increase as expected. Although, the mean scores for STOI PESQ and SDR vary relatively slower with change in SNR, than MI and LLR scores. Further, the variance decreases for STOI and LLR but on contrary it increases for MI SDR and PESQ scores. Similar to CER scores, the effective change is observed at 10 dB SNR. This is an indication of an equivalence between measurement error in microphone position and original signal effected by ambient noise at 10 dB SNR.

4 Conclusions

In this paper, we addressed the impact of error in measuring the position of microphone position on (a) the performance of a multi-channel distant speech recognition in terms of CER and (b) quality and intelligibility of beamformed speech with two well know beamformers, namely, DSB and MVDR. Experimental analysis showed, that with increased

Table 3: Mean (variance) of speech intelligibility and quality for different microphone position measurement errors.

BF	σ^2	Speech Intelligibility		Speech Quality		
		STOI	MI	SDR	PESQ	LLR
DSB	0	0.23 (0.07)	4.72 (2.07)	-18.08 (0.89)	1.48 (0.19)	1.34 (0.29)
	0.01	0.22 (0.07)	4.66 (2.08)	-18.03 (2.18)	1.50 (0.21)	1.24 (0.27)
	0.5	0.17 (0.07)	2.55 (0.81)	-21.70 (2.95)	1.18 (0.22)	3.76 (1.75)
	1	0.17 (0.07)	2.51 (0.83)	-21.70 (3.80)	1.18 (0.22)	3.74 (1.78)
MVDR	0	0.37 (0.34)	22.14 (36.10)	-21.1 (3.75)	1.16 (0.19)	3.82 (1.81)
	0.01	0.37 (0.34)	22.05 (37.03)	-21.3 (3.88)	1.20 (0.22)	3.51 (1.68)
	0.5	0.20 (0.16)	10.55 (36.19)	-21.9 (2.86)	1.20 (0.14)	3.76 (1.74)
	1	0.19 (0.13)	8.42 (30.17)	-21.7 (3.72)	1.17 (0.21)	3.74 (1.81)

Table 4: Variation in speech intelligibility and quality with varying SNRs (in dB)

SNRs	Speech Intelligibility		Speech Quality		
	STOI	MI	SDR	PESQ	LLR
0	0.289 (0.04)	1.57 (0.39)	-21.47 (1.79)	1.25 (0.08)	5.21 (4.34)
5	0.285 (0.03)	1.72 (0.38)	-21.54 (4.17)	1.25 (0.10)	5.02 (4.18)
10	0.291 (0.01)	1.86 (0.41)	-21.40 (6.34)	1.27 (0.13)	4.83 (3.99)
20	0.299 (0.00)	2.11 (0.44)	-21.39 (8.01)	1.28 (0.16)	4.49 (3.65)

measurement error in the location of microphone the quality of mult-channel distant speech recognition deteriorates (higher CER) so does the speech intelligibility and quality, as expected. We further showed that the effect of microphone position measurement error on distant speech recognition in terms of CER is equivalent to a close microphone speech being effected by an additive environmental noise in the range of 5 to 10 dB.

References

Terence Betlehem and Thushara D Abhayapala. 2005. A modal approach to soundfield reproduction in reverberant rooms. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 289–292.

Ritujoy Biswas, Karan Nathwani, and Vinayak Abrol. 2021. Transfer learning for speech intelligibility improvement in noisy environments. *Proc. Interspeech 2021*, pages 176–180.

Xiaoyi Chen, Wenwu Wang, Yingmin Wang, Xionghu Zhong, and Atiyeh Alinaghi. 2015. Reverberant speech separation with probabilistic time–frequency masking for B-format recordings. *Speech Communication*, 68:41–54.

Marco Crocco, Andrea Trucco, Vittorio Murino, and Alessio Del Bue. 2014. Towards fully uncalibrated room reconstruction with sound. In *IEEE European Signal Processing Conference (EUSIPCO)*, pages 910–914.

T. Dietzen, S. Doclo, A. Spriet, W. Tirry, M. Moonen, and T. van Waterschoot. 2017. Low-complexity kalman filter for multi-channel linear-prediction-based blind speech dereverberation. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 284–288.

S. Gannot, D. Burshtein, and E. Weinstein. 2001. Signal enhancement using beamforming and nonstationarity with applications to speech. *IEEE Transactions on Signal Processing*, 49(8):1614–1626.

Sharon Gannot, Emmanuel Vincent, Shmulik Markovich-Golan, and Alexey Ozerov. 2017. A consolidated perspective on multimicrophone speech enhancement and source separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(4):692–730.

Emanuel AP Habets. 2006. Room impulse response generator. *Technische Universiteit Eindhoven, Tech. Rep*, 2(2.4):1.

Jahn Heymann, Lukas Drude, Reinhold Haeb-Umbach, Keisuke Kinoshita, and Tomohiro Nakatani. 2019. Joint optimization of neural network-based wpe dereverberation and acoustic model for robust online asr. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6655–6659. IEEE.

Hugging Face Team. wav2vec2. `https://huggingface.co/transformers/model_doc/wav2vec2.html`.

Hansung Kim, Luca Remaggi, Philip JB Jackson, and Adrian Hilton. 2019. Immersive spatial audio reproduction for vr/ar using room acoustic modelling from 360 images. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 120–126. IEEE.

Alexander Krueger and Reinhold Haeb-Umbach. 2010. Model-based feature enhancement for reverberant speech recognition. *in IEEE Transactions on Audio, Speech, and Language Processing*, 18(7):1692–1707.

K. Kumatani, J. McDonough, S. Schacht, D. Klakow, P. N. Garner, and W. Li. 2008. Filter bank design based on minimization of individual aliasing terms for minimum mutual information subband adaptive beamforming. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1609–1612.

Kenichi Kumatani, Rita Singh, and Bhiksha Raj. 2015. Btk / Millennium ASR Manual. `https://distantspeechrecognition.sourceforge.io/user_doc_btk10.html`.

Li Li and Kazuhito Koishida. 2020. Geometrically constrained independent vector analysis for directional speech enhancement. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 846–850. IEEE.

Philipos C Loizou and Gibak Kim. 2010. Reasons why current speech-enhancement algorithms do not improve speech intelligibility and suggested solutions. *in IEEE Transactions on audio, speech, and language processing*, 19(1):47–56.

Heinrich Löllmann, Andreas Brendel, and Walter Kellermann. 2019. *Comparative study of single-channel algorithms for blind reverberation time estimation*. Universitätsbibliothek der RWTH Aachen.

Alastair H Moore, Mike Brookes, and Patrick A Naylor. 2013. Room geometry estimation from a single channel acoustic impulse response. In *IEEE European Signal Processing Conference (EUSIPCO)*, pages 1–5.

Arulkumaran Muthukumarasamy and Kevin D Donohue. 2009. Impact of microphone placement errors on speech intelligibility. In *IEEE Southeastcon*, pages 323–328.

Karan Nathwani, Morgane Daniel, Gaël Richard, Bertrand David, and Vincent Roussarie. 2016. Formant shifting for speech intelligibility improvement in car noise environment. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5375–5379.

Karan Nathwani, Harish Padaki, and Rajesh M Hegde. 2013. Multi channel reverberant speech enhancement using LP residual cepstrum. In *Asilomar Conference on Signals, Systems and Computers*, pages 555–559.

Karan Nathwani, Gaël Richard, Bertrand David, Pierre Prablanc, and Vincent Roussarie. 2017. Speech intelligibility improvement in car noise environment by voice transformation. *Speech Communication*, 91:17–27.

Patrick A Naylor and Nikolay D Gaubitch. 2010. *Speech Dereverberation*. Springer Science & Business Media.

OpenSLR. 2021. Librispeech ASR corpus. `https://www.openslr.org/resources/12/dev-clean.tar.gz`.

Despoina Pavlidi, Anthony Griffin, Matthieu Puigt, and Athanasios Mouchtaris. 2013. Real-time multiple sound source localization and counting using a circular microphone array. *in IEEE Transactions on Audio, Speech, and Language Processing*, 21(10):2193–2206.

Aditya Raikar, Karan Nathwani, Ashish Panda, and Sunil Kumar Kopparapu. 2020. Effect of microphone position measurement error on RIR and its impact on speech intelligibility and quality. In *Interspeech 2020*, pages 5056–5060.

Joshua M Sachar, Harvey F Silverman, and William R Patterson. 2002. Position calibration of large-aperture microphone arrays. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 1797–1800.

Joshua M Sachar, Harvey F Silverman, and William R Patterson. 2004. Microphone position and gain calibration for a large-aperture microphone array. *IEEE Transactions on Speech and Audio Processing*, 13(1):42–52.

Ofer Schwartz, Sharon Gannot, and Emanuël AP Habets. 2014. Multi-microphone speech dereverberation and noise reduction using relative early transfer functions. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(2):240–251.

Ofer Schwartz, Sharon Gannot, and Emanuël AP Habets. 2015. Nested generalized sidelobe canceller for joint dereverberation and noise reduction. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 106–110.

P. Stoica, Zhisong Wang, and Jian Li. 2002. Robust capon beamforming. In *IEEE Asilomar Conference on Signals, Systems and Computers*, pages 876–880.

I. Szöke, M. Skácel, L. Mošner, J. Paliesek, and J. Černocký. 2019. Building and evaluation of a real room impulse response dataset. *IEEE Journal of Selected Topics in Signal Processing*, 13(4):863–876.

Cees H Taal, Richard C Hendriks, Richard Heusdens, and Jesper Jensen. 2010. A short-time objective intelligibility measure for time-frequency weighted noisy speech. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4214–4217.

J. Taghia and R. Martin. 2014. Objective intelligibility measures based on mutual information for speech subjected to speech enhancement processing. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(1):6–16.

J. Taghia, R. Martin, and R. C. Hendriks. 2012. On mutual information as a measure of speech intelligibility. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 65–68.

Zhenyu Tang, Nicholas J Bryan, Dingzeyu Li, Timothy R Langlois, and Dinesh Manocha. 2020. Scene-aware audio rendering via deep acoustic analysis. *IEEE Transactions on Visualization and Computer Graphics*, 26(5):1991–2001.

Ye yi Wang, Alex Acero, and Ciprian Chelba. 2003. Is word error rate a good indicator for spoken language understanding accuracy. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 57–62.

Yangjie Wei, Ke Zhang, Dan Wu, and Zhongqi Hu. 2021. Exploring conventional enhancement and separation methods for multi-speech enhancement in indoor environments. *Cognitive Computation and Systems*.

Takuya Yoshioka, Armin Sehr, Marc Delcroix, Keisuke Kinoshita, Roland Maas, Tomohiro Nakatani, and Walter Kellermann. 2012. Making machines understand us in reverberant rooms:robustness against reverberation for automatic speech recognition. *IEEE Signal Processing Magazine*, 29(6):114–126.

Wangyang Yu and W Bastiaan Kleijn. 2019. Room geometry estimation from room impulse responses using convolutional neural networks. *arXiv preprint arXiv:1904.00869*.

X. Zhang, Z. Wang, and D. Wang. 2017. A speech enhancement algorithm by iterating single-and multi-microphone processing and its application to robust ASR. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 276–280.

IE-CPS Lexicon: An Automatic Speech Recognition Oriented Indian-English Pronunciation Dictionary

Shelly Jain[*] **Aditya Yadavalli**[*] **Ganesh S Mirishkar**[*]

Chiranjeevi Yarra **Anil Kumar Vuppala**

Speech Processing Laboratory
International Institute of Information Technology, Hyderabad
Gachibowli, Hyderabad, Telangana, 500032
{shelly.jain, aditya.yadavalli, mirishkar.ganesh}@research.iiit.ac.in
{chiranjeevi.yarra, anil.vuppala}@iiit.ac.in

Abstract

Indian English (IE), on the surface, seems quite similar to standard English. However, closer observation shows that it has actually been influenced by the surrounding vernacular languages at several levels from phonology to vocabulary and syntax. Due to this, automatic speech recognition (ASR) systems developed for American or British varieties of English result in poor performance on Indian English data. The most prominent feature of Indian English is the characteristic pronunciation of the speakers. The systems are unable to learn these acoustic variations while modelling and cannot parse the non-standard articulation of non-native speakers. For this purpose, we propose a new phone dictionary developed based on the Indian language Common Phone Set (CPS). The dictionary maps the phone set of American English to existing Indian phones based on perceptual similarity. This dictionary is named Indian English Common Phone Set (IE-CPS). Using this, we build an Indian English ASR system and compare its performance with an American English ASR system on speech data of both varieties of English. Our experiments on the IE-CPS show that it is quite effective at modelling the pronunciation of the average speaker of Indian English. ASR systems trained on Indian English data perform much better when modelled using IE-CPS, achieving a reduction in the word error rate (WER) of upto 3.95% when used in place of CMUdict. This shows the need for a different lexicon for Indian English.

1 Introduction

Today, speech processing technology is gaining an undeniable importance. Automatic speech recognition (ASR) systems in particular are being sought after, as smart assistants like Siri, Alexa and Google Assistant grow in popularity at remarkable rates. Effective and accurate speech recognition is also a major concern in India for two major reasons – firstly, India is home to thousands of spoken languages, not all of which possess written forms; secondly, India has a low literacy rate, so a considerable portion of the population can communicate only via speech. Unfortunately, there is a dearth of available speech data on many languages spoken in India, so most ASR systems tend to be ineffective at modelling these. In order to both overcome the drawbacks caused by lack of available speech data, and leverage the similarities between the various Indian languages, linguistically motivated approaches are gaining appreciation among researchers in the field.

Many research groups have started working towards having a good speech recognition system for all the major Indian languages. To achieve this, it is necessary to have a good pronunciation modelling block, consisting of a grapheme-to-phoneme (G2P) system. A G2P is needed by Text-to-Speech (TTS) systems as well. Nair et al. (2013) work on building a rule-based G2P on the low-resource Malayalam language. Parlikar et al. (2016) work on building a G2P for major Indian languages such as Hindi, Tamil and Telugu and test it on TTS systems. Mortensen et al. (2018) build a multilingual G2P that works for 61 languages out of the box, in which 7 Indian languages are supported. They also provide a guideline to integrate any language into their system. Arora et al. (2020) work on statistical G2P that predicts schwa deletion in Hindi more accurately and show improvements over their baseline models. Wasala et al. (2006) work on Sinhala

[*]These authors contributed equally to the paper

G2P, a language that is closely related to other Indian languages, and further, propose rules to handle schwa epenthesis.

English is also a language with widespread use in India. Given India's rich language diversity, English has largely been adopted in the spheres of the educated public as a *lingua franca* (Ng and Hirose, 2012; Kim et al., 2011) like in other parts of the world. The English spoken in India is influenced by the other languages with which it exists in constant contact, affecting its structure and vocabulary as well as its pronunciation. This means that existing ASR systems for General American English (GAE) or British English (BrE) (Ferragne and Pellegrino, 2004) are markedly ineffective when dealing with Indian English (IE). New ASR systems need to be created which are capable of processing the variety of English spoken in India, catering to the accents and language patterns (Jain et al., 2018) of the people speaking it.

Considering the limited work in this direction and the impact it may have, we study the major phonetic variation between Indian English and standard English, and propose an Indian English lexicon which can be used in an IE ASR system. To show its effectiveness, the performance of the two different ASR systems is compared on utterances of Indian English – an American English ASR system trained on Librispeech (Panayotov et al., 2015) and an Indian English ASR system trained on National Programme on Technology Enhanced Learning (NPTEL) speech corpus (described further in Section 4.1).

The organisation of the remainder of the paper is as follows. Section 2 describes prior work. Section 3 details the proposed lexicon and the rationale behind the chosen mappings. Sections 4 and 5 explain the experimental setup and results when using the new dictionary for ASR. Further discussion on the need for this lexicon is done in Section 6. Finally, possible future ventures are briefly mentioned in Section 7, and the paper is concluded in Section 8.

2 Related Work

ARPAbet is a phonetic transcription code used to represent the phonemes and allophones of GAE using distinct ASCII sequences. The CMU Pronouncing Dictionary (CMUdict) (Weide, 2005) is an open-source pronunciation dictionary which makes use of a modified form of ARPAbet. Currently, there has been very limited work done to replicate such efforts for other languages, or even other varieties of English. Our proposed lexicon attempts to fill this gap in the field of language phonology.

The work by Ganji et al. (2019) is also closely related to our line of research in this paper. They collected a Hindi-English code-switched speech corpus. In addition to releasing that corpus, they proposed a few measures that could help model such speech better. One of the measures they proposed was a way to handle the pronunciation model block for this code-switched corpus that is most often used in ASR and text-to-speech (TTS) systems. They proposed a CMUdict-based phone mapping with some of the English phones mapped to a common Indian language phone set (Ramani et al., 2013) based on perceptual similarity. The paper, however, did not present any experimental results for ASR with their proposed pronunciation model. We explain how we build upon this in greater detail, in Section 3.

Another study by Huang et al. (2020) compared the phonology of GAE and the phonology of (Hindi-based) IE and developed a phoneme set for IE in X-SAMPA (Wells, 1995) similar to that of GAE. However, the dataset on which they built the ASR system has its own limitations. It has speech data from just 10 native Hindi speakers with long-term exposure to only the English spoken in New Delhi. As such, the language modelled was a very specific variety of IE, which is not representative of a large-scale Indian population.

Yarra et al. (2019) collected a corpus of Indian English, consisting of 240 hours of recorded speech. This is called Indic TIMIT. They extensively analysed Indian English accent and compared it to standard English. They further came up with various kinds of rules to improve the pronunciation model for Indian English speech. However, such rules needed another complex module – a letter to phoneme aligner. The letter to phoneme aligner is used to align a word's letters to its individual sound units. Our work eliminates the need for this module, by applying only those rules which were exhibited by native speakers of a majority of Indian languages. In addition to that, Yarra et al. (2019) also did not use the now largely standardised phone codes for six major Indian languages – Hindi, Marathi, Bengali, Tamil, Malayalam and Telugu – proposed by Ramani et al. (2013), the Common Phone Set (CPS). This makes Yarra et al.

(2019)'s work difficult to use in a multilingual or a code-switched setting where one of the languages is an Indian language. They also did not compare the performance of their proposed-lexicon-based ASR system and CMUdict-based ASR system. Therefore, it is unclear what benefits their proposed lexicon might have to an ASR system.

In addition to the above works, the studies by Schilk (2010); Bansal (1969); Mohan (2021) were helpful in providing the general perspective on Indian English. They described some specific differences made in the pronunciation of English by Indians, at both the sound and word level. We incorporate some of the differences they point out at the sound level in our lexicon. In addition to the differences, they also provided insight into the question of whether Indian English is better treated as a variety of standard English or as a completely separate language. This is discussed further in Section 6.

3 Indian English-Common Phone Set (IE-CPS)

Given the limited availability of annotated IE speech data, there is need for a mechanism to provide the pronunciations for this kind of speech. Our lexicon, Indian English Common Phone Set (IE-CPS), is able to provide a simple yet effective solution to this by leveraging the existing CMUdict, avoiding use of large amounts of data and complex architectures to model IE speech.

All of the English phones in CMUdict have been mapped to the existing Common Phone Set (CPS) phones proposed by Ramani et al. (2013) (Black et al., 1998). CPS is a language-independent set of the phones of six major Indian languages (Hindi, Marathi, Bengali, Tamil, Malayalam and Telugu) which maps parallel phones to the same code. Reusing CPS was favoured over creating a new encoding because it already provides a phone dictionary for a large set of Indian languages, and is continuously extended to include a greater number. Since English, too, is becoming more widespread in India, extending CPS to include the proposed Indian English phone dictionary is the logical consequence. While some of the mappings in this paper are inspired by Ganji et al. (2019), we did make changes to further reflect the accent that native Indian language speakers would have when speaking English. These changes are elaborated on in Section 3.2.

3.1 Lexicon Construction

Indian English differs notably from standard American or British English. IE speech also exhibits further variation according to the other languages prevalent in each region IE-CPS models only those sound transformations which can be observed among a majority of the different linguistic regions. The first of these is the absence of dental fricatives ([ð], [θ]) and alveolar plosives ([d], [t]) in Indian languages, replaced by dental plosives ([d̪], [t̪ʰ]) and retroflex plosives ([ɖ], [ʈ]) respectively. The second notable distinction is the use of long vowels in place of diphthongs (barring two cases – [aʊ] and [aɪ]). The third is the lack of distinction between the sounds [v] and [w] in Indian languages. The final difference is the approximation of vowel sounds to the nearest vowel in the speakers' native language. Additional differences between American English and Indian English, such as the prosodic features of GAE and the phonemic aspiration in IE, do not exist as differences at the phonetic level; hence these are not reflected in the proposed lexicon.

As Yarra et al. (2019) have already pointed out, Indians speaking English do not add or delete phonemes as often as they substitute them. Our independent analysis also leads to similar numbers to Yarra et al. (2019)'s work. While deletions happen very rarely ($< 1\%$), insertions happen a little more often ($< 15\%$). However, substitutions occur frequently when Indians speak English ($> 30\%$)[1]. Later in this paper, we show that just substituting phones causes a drastic drop in WER. IE-CPS describes these substitutions, but has no rules for insertion or deletion. The advantage of such an approach is the ability to remove the letter to phoneme alignment module, which considerably simplifies the pronunciation model.

We also noticed that the behaviours like insertions differ significantly based on regional accent – for example, frequent vowel epenthesis by speakers of Hindi. Since such sound changes are not common among all accents, these rules are required only when modelling specific regional accents. Thus, we have not included rules of phone insertion in our lexicon. Additionally, this also simplifies the use of the IE-CPS in multilingual data containing English – IE-CPS can easily be applied to each variety of Indian English since accent specific behaviour is not replicated.

[1]For the remaining pronunciations, no errors were observed.

3.2 Indian English Common Phone Set

Based on previously mentioned observations, the changes made to the CMU pronunciation dictionary are broadly classified into four categories:

- **Mapping of English phones to existing CPS codes.** Ganji et al. (2019) introduce new codes for phones `AA` and `AO`, i.e., `ao`. The IE-CPS maps these phones to the existing CPS code, `ou`. We chose this specific CPS code because the phone that it denotes is perceptually similar, being the closest phone in the Indian phone inventory. This has the advantage of minimising the need for introducing new phonetic codes. Along with these, various other consonant and vowel sounds have been mapped to codes in CPS which are perceptually similar but not the same phone.

- **Phones with multiple mappings.** Some examples are `Z` and `ZH`. `Z` gets mapped to `z` and `j`. Similarly, `ZH` gets mapped to three different CPS codes – `z`, `j` and `jhq`. This is because while many Indian languages exhibit free variation between the sounds of `j` and `z`, the increasing exposure of Indians to both American and British varieties of English is resulting in a new phone (represented by `jhq`) being gradually realised as a separate phoneme. The use of multiple allophones is absent in the codes proposed by Ganji et al. (2019), and both `Z` and `ZH` are mapped to `z`.

- **Addition of new mappings.** Huang et al. (2020); Yarra et al. (2019) explain how the diphthongs in English (barring `ai` and `au`) are pronounced as a single long vowel when Indians speak English. To reflect this, we map `EY` and `OW` to existing CPS codes `ee` and `oo` respectively, which represent long utterances of the first vowels in their corresponding diphthongs. This is in contrast to the use of `ei` and `o` respectively by Ganji et al. (2019), which are codes for shorter vowel utterances.

- **Exclusion of some CPS mappings.** English does not have phones analogous to all the CPS codes in its phoneme inventory. Those absent in Indian English (like the phones corresponding to `q`, `txh`, `lx`, etc) have thus not been included in the proposed lexicon.

Example	CMUdict		IE-CPS	
	ARPAbet	IPA	Code	IPA
odd	AA	ɑ	ou[2]	ɔ
at	AE	æ	ae	æ
hut	AH	ʌ	a	a
ought	AO	ɔ	ou	ɔ
cow	AW	aʊ	au	aʊ
hide	AY	aɪ	ai	aɪ
be	B	b	b	b
cheese	CH	ʧ	c	ʧ
dark	D	d	dx	ɖ
this	DH	ð	d	d̪
egg	EH	ɛ	e[3]	ɛ
hurt	ER	ɝ	**er**	əː
wait	EY	eɪ	ee	eː
fin	F	f	f	f
game	G	g	g	g
hill	HH	h	h	ɦ
sit	IH	ɪ	i	ɪ
eat	IY	i	ii	iː
joke	JH	ʤ	j	ʤ
key	K	k	k	k
let	L	l	l	l
map	M	m	m	m
note	N	n	n	n
sing	NG	ŋ	ng	ŋ
boat	OW	oʊ	oo	oː
toy	OY	ɔi	**oy**	ɔi
pen	P	p	p	p
read	R	ɹ	r	r
sea	S	s	s	s
show	SH	ʃ	sh	ʃ
tea	T	t	tx	ʈ
thin	TH	θ	th	t̪ʰ
put	UH	ʊ	u	u
food	UW	u	uu	uː
villa	V	v	w	ʋ
we	W	w	w	ʋ
yes	Y	j	y	j
zip	Z	z	z	z
zip	Z	z	j	ʤ
seizure	ZH	ʒ	jhq[4]	ʒ
seizure	ZH	ʒ	z	z
seizure	ZH	ʒ	j	ʤ

Table 1: The proposed Indian English Common Phone Set (IE-CPS). The IE-CPS codes marked in **bold** in column 4 indicate the phones unique to Indian English.

The complete set of Indian English phones is given in Table 1. Each of the phones of GAE, as provided by CMUdict, has been mapped to existing phones from CPS. In cases where this was not possible, additional phones unique to the Indian English phone set have been added to CPS. This lexicon was verified by a trained linguist specialised in the study of Indian languages.

Word	CMUdict	IE-CPS
thought	`TH AO T`	`th ou tx`
waited	`W EY T IH D`	`w ee tx i dx`

Table 2: Comparison of CMUdict and IE-CPS with example words.

In Table 2, we show the phone break down of two examples in the case of CMUdict and proposed dictionary. In the examples taken, i.e, the words 'waited' and 'thought', one can notice the following differences:

- **Shift in place of articulation.** This change can be seen in several places in the examples taken. In both the example words, the code `T` in CMUdict is mapped to `tx` in IE-CPS. `tx` denotes a retroflex plosive whereas in the case of CMUdict the place of articulation is alveolar. As the latter is absent from Indian languages, the closest phoneme (here, `tx`) is substituted. Similar substitution is seen for the CMUdict codes `TH`, `W` and `D` (replaced by `th`, `w` and `dx` respectively).

- **Diphthong to long vowel.** As already mentioned, the diphthong in "waited" is pronounced as a long vowel when Indians speak English. The same is encoded when the code `EY` in CMUdict is mapped to `ee` in IE-CPS

Note here that not all sounds are changed in the Indian pronunciation of the words. The codes `AO` (CMUdict) and `ou` (IE-CPS) both represent the same phone [ɔ]. Hence, the vowel sound in 'thought' is the same in both American and Indian varieties of English.

4 Experimental Setup

In this subsection, we elaborate on the GAE and IE speech corpora upon which we ran our experiments. We also briefly describe two components of the ASR system – the acoustic model and the language model. These components have not been changed in the study, and are only used to highlight the effectiveness of the IE-CPS as a lexicon for Indian English ASR.

4.1 Corpora

The experiments are conducted on the following datasets. The details about the datasets are explained below:

1. **NPTEL Data**: National Programme on Technology Enhanced Learning (NPTEL)[5] is an open-source e-learning platform which is mainly maintained and organised by top-tier academic institutes from India (like Indian Institute of Technology (IIT) and Indian Institute of Science (IISc)). It covers a wide range of courses, including engineering, basic sciences, management, law, and personality development. As part of the challenge conducted by IIT Madras, the organisers released 80 hours of read speech and 200 hours of recorded lectures (on Computer Science and Electrical Engineering) from NPTEL[6]. The test set contains 12 hours of read speech and 10 hours of lectures (spontaneous speech). We report the results on both kinds of speech separately in Section 5. All of the audio is sampled at 16Khz. There is no explicit speaker information given because the dataset was shared by the organisers of the ASR challenge. However, we do confirm that all of the recorded speech is in Indian English. The speech is well distributed among several regions and diverse accents, thus preventing an issue of overfitting on a single variety of Indian English.

2. **Librispeech**: This is a corpus of approximately 1000 hours of 16Khz American English[7] read speech. The dataset is derived from the audiobooks that are part of the Librivox project. We take 960 hours of the corpus

[2]In CPS, this code is mapped to [oʊ] but the character this code represents is also allophonic to [ɔ]. Hence it is used in the latter sense here.

[3]Similar to the previous case, the character associated with this code is mapped to the allophones [e] and [ɛ], the latter of which is used here.

[4]Officially this is included in CPS, but in practical use the character representing this phone is never seen in text, or is pronounced the same as `jh` ([dʒʰ]).

[5]https://nptel.ac.in/

[6]https://sites.google.com/view/englishasrchallenge/home

[7]Strictly speaking this is not fully American English but a mix. However, most of the audiobooks part of the dataset use American English.

for training the model and 5 hours for testing it. There are no overlapping speakers in the train and test sets. There are a total of 1166 speakers in this dataset. The dataset is gender balanced to ensure that there is no bias towards one gender.

4.2 Acoustic Model

The acoustic model (AM) includes a Gaussian Mixture Model-Hidden Markov Model (GMM-HMM), a hybrid Deep Neural Network (DNN-HMM), and a Time Delay Neural Network (TDNN) (Peddinti et al., 2015).

In this work, Mel-frequency cepstral coefficients (MFCC) features, computed for a 20 ms window with 10 ms overlap, are fed into the system for initial speaker-independent GMM-HMM training. The speaker-dependent GMM-HMM model is built using Feature space Maximum Likelihood Linear Regression (fMLLR) features (Gales, 1998). For DNN-based acoustic model training, we use three hidden layers. Each hidden layer contains 2000 dimensional hidden units with p-norm activation. As the input p-norm dimension is 2000, the resultant output p-norm dimension is of 400 dimensions ($p = 2$ and group size=5). The DNN has an input layer that takes 360-dimensional input, and the DNN's output is 2365 context-dependent phonemes states.

The outputs are obtained by GMM-HMM alignment. The cross-entropy loss is minimised by using back-propagation with an initial learning rate of 0.01 and a final learning rate of 0.001. In addition to these 39-dimensional MFCCs, 100-dimensional iVectors (Dehak et al., 2010) are appended at each time step. It has been noted that iVectors capture both speaker and environment-specific information and are useful for rapid and discriminative adaptation of the neural network. The training data was increased 3-fold artificially through time-warping of raw audio (Ko et al., 2015). The training procedure for chain models is a Lattice-Free (LF) version of the Maximum Mutual Information (MMI) (Povey et al., 2016) criterion without the need for frame-level cross-entropy pre-training.

We use the chain TDNN model comprising 6 layers and 725 Rectified Linear Units (ReLU) at the input layer. The input features are at the original frame rate of 100 per second, and the output frame is reduced by 3 times. The first splicing is removed before the Linear Discriminant Analysis (LDA) transformation layer. The spliced indices in the consecutive layers were $[-1, 0, 1; -3, 0, 3; -3, 0, 3; -3, 0, 3; -6, -3, 0]$ with LDA applied to the input features.

4.3 Language Model

The Language Model (LM) predicts the probability of a hypothesised word sequence by learning the correlation between words for given text corpora. In this study, the text data corresponding to the respective audio is normalised. The SRI Language modelling toolkit (Stolcke, 2002) is used to train Kneser-Ney (Chen and Goodman, 1999) smoothed tri-gram LM on the text corpora. No external text is used.

All the experiments (both acoustic and language modelling) have been conducted using the Kaldi speech recognition toolkit (Povey et al., 2011). The experiments on Librispeech were conducted using the standard Kaldi s5 recipe[8]. The same recipe was adapted to run experiments on the NPTEL data. The training was carried out using GeForceGTX 2080 Ti. The metric used for evaluating the ASR system's performance is word error rate (WER).

5 Results

This section presents our results and analysis on IE-CPS for acoustic model training. Further, we compare and analyse the scenarios in which it outperforms the existing CMUdict. All the experiments were conducted on Librispeech and NPTEL data as mentioned in Section 4.1. For evaluating IE-CPS on a speech recognition task, two sets of experiments were performed with varying conditions, i.e., read and spontaneous condition.

5.1 Experiments with CMUdict

In the first study, the acoustic model is trained using CMUdict on both the speech corpora. The obtained results are shown in Table 3. As expected, the performance of the model trained on the Librispeech corpus was much better than the performance of the model trained on the NPTEL database. This held true for all three architectures of the acoustic model. The primary reason for this is that Librispeech is oriented to the American accent, which was the reference for the pronunciation dictionary.

[8] https://github.com/kaldi-asr/kaldi/blob/master/egs/librispeech/s5/run.sh

Corpus	Training	Testing	Acoustic Model	WER (%)	
				CMUdict	IE-CPS
Librispeech	Read	Read	GMM	7.02	16.72
			DNN	4.90	15.85
			TDNN	4.43	13.08
NPTEL	Read + Lecture (Spontaneous)	Read	GMM	10.64	**6.52**
			DNN	8.25	**4.89**
			TDNN	6.67	**3.76**
		Lecture	GMM	12.45	**8.12**
			DNN	10.97	**6.48**
			TDNN	8.85	**4.90**

Table 3: Comparison of results using CMUdict and IE-CPS.

5.2 Experiments with IE-CPS

In this study, the ASR pipeline uses the proposed IE-CPS lexicon instead of the CMUdict. The WER (%) for the same is shown in Table 3. The use of the IE-CPS improves the performance of the ASR trained on NPTEL (Indian-English) data. On the other hand, it performs worse than CMUdict when the ASR is trained on Librispeech. This further demonstrates the significant difference between the Indian and American varieties of English. Table 4 shows an example of how the ASR trained on IE data has poorer performance when modelled using CMUdict, a lexicon not specifically designed for it. The highlighted mistake made with CMUdict is corrected when using IE-CPS, as a direct result of applying substitution rules with proper linguistic foundation.

In the example provided in Table 4, [ð] (from 'th' in 'this') is realised as [d̪] in IE speech due to their phonetic similarity. However, [d̪] is also phonetically similar to [d] (from 'd' in 'disappointing'). This poorly defined contrast between [ð] and [d] adversely affects the LM when the ASR is modelled using CMUdict, but since the distinction is clear in the IE-CPS mappings, the ASR using IE-CPS is successful at disambiguating such cases.

6 Discussion & Analysis

In this section we investigate the behaviour of the IE-CPS in more detail by asking the questions detailed below.

If Indian English varies so much with region, then how can this common lexicon be useful? It has been mentioned earlier that the regional languages have a significant effect on spoken English. This results in several different accents and thus phoneme inventories. However, IE-CPS is based on the general speech characteristics observed in speakers of a range of languages. The rules applied to map the phones from American to Indian English are generated from the most prominent and widespread approximations made by Indians speaking English, such as the transformation from alveolar to retroflex plosives. The notable improvement demonstrated by experimental results performed on the NPTEL speech data (consisting of a broad range of regional Indian accents) also proves the overall effectiveness of the lexicon despite the variation in Indian English.

Consequently, it may be extrapolated that such a lexicon can be applied to the English in regions where academic effort to model the speech is still deficient – such as North-East India, with languages Bodo, Khasi, Assamiya, Meitei, Mizo – without any extra linguistic expertise on the matter. Any required region specific tuning also becomes easier as there are fewer changes to make, which can be done using the already mostly standardised encoding of CPS. Furthermore, if there is the need for a lexicon for multilingual speech (also containing English), such a pronunciation dictionary with the phones common to most varieties of Indian English would serve to better represent the multiple possible accents.

End-to-End models do not need a lexicon. What is the need for such a lexicon? With sufficiently large amounts of data, it is possible to discard pronunciation models entirely and build end-to-end (E2E) ASR systems. However, as it

Lexicon	Sentence
IE-CPS	**Its disappointing** when you are convinced that there is nothing
CMUdict	**This appointing** when you are convinced that there is nothing
Ground Truth	Its disappointing when you are convinced that there is nothing

Table 4: Comparison of Indian English ASR system when using IE-CPS and CMUdict. The words in **bold** were the ones confused in the latter case.

is with many Indian languages, if the language (or the accented language in this case) is low resource then E2E models would not be a good solution. In such cases, conventional hybrid systems perform better. Hybrid systems require us to explicitly model the pronunciation of the language, i.e., a mapping from grapheme to phoneme (G2P) is required. While one can model this with sequence-to-sequence (seq2seq) architectures (Gorman et al., 2020; Peters et al., 2017), even that would require relatively large amounts of G2P data. In cases where even that is not available, rule-based parsers are the only choice for researchers wanting to model a language or an accent. Additionally, even with large training data, seq2seq models the *long tail* in the distribution of the data causes a deterioration in the overall performance. Rule-based parsers avoid this issue as they do not rely on learning the data. In such scenarios, the rule-based IE-CPS can be used to obtain considerable improvements as shown in Table 3. In fact, the lexicon can directly be substituted in place of another lexicon in an existing ASR pipeline, making its integration simpler. The other advantages of this lexicon are that, being completely rule-based, it would require no training data or additional computational resources as opposed to seq2seq-based G2P systems.

7 Future Work

The current IE-CPS generalises several characteristics of spoken Indian English. However, our preliminary analysis reveals that fine-tuning the proposed lexicon to a particular regional accent of Indian English, such as Bengali or Malayalam, results in a better ASR performance when testing on the same regional accent. For example, a Bengali accent would cause [a] to be pronounced as [ɔ], and aspiration of consonants would be largely absent for a Malayalam speaker. Such specific changes can further be adapted in IE-CPS to reflect regional varieties of Indian English. Therefore, we plan to propose different lexicons for different regional accents of Indian English in the future.

There is a growing amount of interest in modelling code-switched speech, especially in the Indian community (Manghat et al., 2020). This can be seen by how successful the code-switching subtask was in the 2021 *Multilingual and code-switching (MUCS)* ASR challenges for low resource Indian languages[9] (Diwan et al., 2021). We also noticed that there is an increased interest in having a specialised pronunciation block for such settings. In this direction, we plan to use IE-CPS to model the Indian English part of the code-switched speech. This is especially useful because the other Indian language can be modelled with the same set of phone codes – CPS. Therefore, such a lexicon would be easy to integrate into existing frameworks in such settings.

8 Conclusions

In this paper, we propose a lexicon designed for use in ASR – the Indian English Common Phone Set (IE-CPS). The IE-CPS is a lexicon that can be easily integrated into the existing, (largely) standardised phonetic code for Indian languages – the Common Phone Set (CPS). We show the improvements in an Indian English ASR system when IE-CPS is used as the pronunciation model. This proves that fine-tuning the pronunciation model to Indian English when the ASR system is deployed to work on Indian English speech is the correct way going forward. This is especially true when the user is limited by either the available data or the computational resources to utilise the data.

Acknowledgements

We would like to thank Dipti Misra Sharma for her assistance in verifying and validating this IE-CPS lexicon.

[9]https://navana-tech.github.io/IS21SS-indicASRchallenge/

References

Aryaman Arora, Luke Gessler, and Nathan Schneider. 2020. Supervised grapheme-to-phoneme conversion of orthographic schwas in hindi and punjabi. *ArXiv*, abs/2004.10353.

R. K. Bansal. 1969. *The intelligibility of Indian English: measurements of the intelligibility of connected speech, and sentence and word material, presented to listeners of different nationalities*. Central Institute of English.

Alan W Black, Kevin Lenzo, and Vincent Pagel. 1998. Issues in building general letter to sound rules. In *The third ESCA/COCOSDA workshop (ETRW) on speech synthesis*.

Stanley F Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394.

Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. 2010. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798.

Anuj Diwan, Rakesh Vaideeswaran, Sanket Shah, Ankita Singh, Srinivasa Raghavan K. M., Shreya Khare, Vinit Unni, Saurabh Vyas, Akash Rajpuria, Chiranjeevi Yarra, Ashish R. Mittal, Prasanta Kumar Ghosh, Preethi Jyothi, Kalika Bali, Vivek Seshadri, Sunayana Sitaram, Samarth Bharadwaj, Jai Nanavati, Raoul Nanavati, Karthik Sankaranarayanan, Tejaswi Seeram, and Basil Abraham. 2021. Multilingual and code-switching ASR challenges for low resource indian languages. *CoRR*, abs/2104.00235.

Emmanuel Ferragne and François Pellegrino. 2004. A comparative account of the suprasegmental and rhythmic features of british english dialects. *Modelisations pour l'Identification des Langues*.

Mark JF Gales. 1998. Maximum likelihood linear transformations for hmm-based speech recognition. *Computer speech & language*, 12(2):75–98.

Sreeram Ganji, Kunal Dhawan, and Rohit Sinha. 2019. Iitg-hingcos corpus: A hinglish code-switching database for automatic speech recognition. *Speech Communication*, 110:76–89.

Kyle Gorman, Lucas F.E. Ashby, Aaron Goyzueta, Arya McCarthy, Shijie Wu, and Daniel You. 2020. The SIGMORPHON 2020 shared task on multilingual grapheme-to-phoneme conversion. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 40–50, Online. Association for Computational Linguistics.

Xian Huang, Xin Jin, Qike Li, and Keliang Zhang. 2020. On construction of the asr-oriented indian english pronunciation dictionary. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 6593–6598.

Abhinav Jain, Minali Upreti, and Preethi Jyothi. 2018. Improved accented speech recognition using accent embeddings and multi-task learning. In *Interspeech*, pages 2454–2458.

Sunhee Kim, Kyuwhan Lee, and Minhwa Chung. 2011. A corpus-based study of english pronunciation variations. In *Twelfth Annual Conference of the International Speech Communication Association*.

Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. 2015. Audio augmentation for speech recognition. In *Proc. Interspeech*.

Sreeja Manghat, Sreeram Manghat, and Tanja Schultz. 2020. Malayalam-English Code-Switched: Grapheme to Phoneme System. In *Proc. Interspeech 2020*, pages 4133–4137.

Peggy Mohan. 2021. *WANDERERS, KINGS, MERCHANTS: the story of india through its languages*. PENGUIN.

David R Mortensen, Siddharth Dalmia, and Patrick Littell. 2018. Epitran: Precision g2p for many languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

S. Nair, Cr Rechitha, and C. S. Kumar. 2013. Rule-based grapheme to phoneme converter for malayalam.

Raymond WM Ng and Keikichi Hirose. 2012. Syllable: A self-contained unit to model pronunciation variation. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4457–4460. IEEE.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210.

Alok Parlikar, Sunayana Sitaram, Andrew Wilkinson, and Alan W Black. 2016. The festvox indic frontend for grapheme to phoneme conversion. In *WILDRE: Workshop on Indian Language Data-Resources and Evaluation*.

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*.

Ben Peters, Jon Dehdari, and Josef van Genabith. 2017. Massively multilingual neural grapheme-to-phoneme conversion. *arXiv preprint arXiv:1708.01464*.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. 2011. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, CONF. IEEE Signal Processing Society.

Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur. 2016. Purely sequence-trained neural networks for asr based on lattice-free mmi. In *Proc. Interspeech*, pages 2751–2755.

B Ramani, S Lilly Christina, G Anushiya Rachel, V Sherlin Solomi, Mahesh Kumar Nandwana, Anusha Prakash, S Aswin Shanmugam, Raghava Krishnan, S Kishore Prahalad, K Samudravijaya, et al. 2013. A common attribute based unified hts framework for speech synthesis in indian languages. In *Eighth ISCA Workshop on Speech Synthesis*.

Marco Schilk. 2010. Pingali sailaja, indian english (dialects of english). edinburgh: Edinburgh university press, 2009. pp x 172. hardback £60.00, isbn 978-0-7486-2594-9, paperback £19.99, isbn 978-0-7486-2595-6. *English Language and Linguistics*, 14(1):135–139.

Andreas Stolcke. 2002. Srilm-an extensible language modeling toolkit. In *Proc. Seventh International Conference on Spoken language Processing*.

A. Wasala, R. Weerasinghe, and K. Gamage. 2006. Sinhala grapheme-to-phoneme conversion and rules for schwa epenthesis. In *ACL*.

Robert Weide. 2005. The carnegie mellon pronouncing dictionary [cmudict. 0.6]. *Pittsburgh, PA: Carnegie Mellon University*.

J. Wells. 1995. Computer-coding the ipa: a proposed extension of sampa.

Chiranjeevi Yarra, Ritu Aggarwal, Avni Rajpal, and Prasanta Kumar Ghosh. 2019. Indic timit and indic english lexicon: A speech database of indian speakers using timit stimuli and a lexicon from their mispronunciations. In *2019 22nd Conference of the Oriental COCOSDA International Committee for the Co-ordination and Standardisation of Speech Databases and Assessment Techniques (O-COCOSDA)*, pages 1–6.

An Investigation of Hybrid architectures for Low Resource Multilingual Speech Recognition system in Indian context

Ganesh S Mirishkar **Aditya Yadavalli** **Anil Kumar Vuppala**
Speech Processing Laboratory,
Language Technologies Research Centre,
Kohli Center on Intelligent Systems,
International Institute of Information Technology, Hyderabad
India
{mirishkar.ganesh, aditya.yadavalli}@research.iiit.ac.in,
anil.vuppala@iiit.ac.in

Abstract

India is a land of language diversity. There are approximately 2000 languages spoken around, and among which officially registered are 23. In those, there are very few with Automatic Speech Recognition (ASR) capability. The reason for this is the fact that building an ASR system requires thousands of hours of annotated speech data, a vast amount of text, and a lexicon that can span all the words in the languages. At the same time, it is observed that Indian languages share a common phonetic base. In this work, we build a multilingual speech recognition system for low-resource languages by leveraging the shared phonetic space. Deep Neural architectures play a vital role in improving the performance of low-resource ASR systems. The typical strategy used to train the multilingual acoustic model is merging various languages as a unified group. In this paper, the speech recognition system is built using six Indian languages, namely Gujarati, Hindi, Marathi, Odia, Tamil, and Telugu. Various state-of-the-art experiments were performed using different acoustic modeling and language modeling techniques.

1 Introduction

According to the 2011 census, India has 23 constitutionally recognized official languages and 1600 other languages (Wikipedia, 2021). In this era of digitization, speech technologies for Indian languages play a pivotal role across various business domains. Building automatic speech recognition (ASR) (Reddy, 1976) and text-to-speech (TTS) (Dutoit, 1997) based interfaces for Indian markets is big challenge as majority of languages are "low resourced" (Miao et al., 2013). In general, a language is referred to as low resource when there is: (i) lack of availability of speech, text, transcribed data, (ii) lack of linguistic expertise in a particular language, or (iii) lack of pronunciation dictionary (Lu et al., 2013). In order to build state-of-the-art ASR systems for Indian languages, tons of training data is required for achieving human parity. In the training ASR system, the model expects audio data and corresponding transcripts as an input. Though there is a lot of raw speech freely available for Indian languages, it is a complex and costly process to get the corresponding transcription. Hence, very few efforts were made in these directions over the past decade.

As Indian languages are syllabic, efforts were put in generating the pronunciation dictionary from a simple rule-based parser (Prahallad et al., 2012; Baby et al., 2016; Ramani et al., 2013; Pandey et al., 2017). Indian languages have a peculiar attribute of sharing the same phonetic space. However, they differ phonotactically[1] (Prahallad et al., 2012). So these attributes could be exploited to build an ASR system for achieving better performance. With the introduction of Digital India Mission in 2015, there have been efforts towards handling these low resource languages for building speech recognition technologies.

Different approaches have been proposed in acoustic modeling over the recent years to address the low resource speech recognition problem. In (Swietojanski et al., 2012), the attempt has been made to use cross-lingual acoustic data to initialize deep neural network (DNN) based acoustic models through unsupervised restricted Boltzmann machine (RBM) pre-training. It showed that unsupervised pre-training remains vital for the hybrid setups, especially with limited amounts of transcribed training data. The idea of transfer learning approach was introduced for handling low resource languages in (Imankulova et al., 2019; Cho et al., 2018; Das and Hasegawa-Johnson, 2015). Data augmentation approaches proposed in (Liu et al.,

[1] In general phonotactic is defined as the study of the rules governing the possible phoneme sequences in a language.

2019; Thomas et al., 2020; Cui et al., 2015) were effective in low resource settings.

A different line of work (Chuangsuwanich, 2016; Thomas et al., 2012; Rahimi et al., 2019; Xu et al., 2015), where attempts have been made to extract multilingual features that help in improving low resource ASR systems started gaining prominence. In practice, a language identification (LID) block is used a front-end wherein it tries to predict the language first, and later it is mapped to the corresponding monolingual system or the best possible monolingual system. In this type of approach, the LID block, which acts as a front-end, should be as accurate as possible and robust enough to operate in a multi-thread environment. So to circumvent this issue, acoustic models where it handles multiple languages without any prior knowledge of the language have been proposed (Vydana et al., 2018). In (Shetty and Umesh, 2021), authors have explored the benefits of phonetic sound principles and treated each character unit across the languages as a separate entities with the help of Common Label Set (CLS) approach.

In this paper, the authors compare and analyze different Time Delay Neural Networks (TDNN) variants (Sugiyama et al., 1991) as they seem to be best suited for such kind of low resource speech recognition tasks. This paper investigates the effectiveness of different acoustic models for low resource multilingual speech recognition for six Indian languages (namely Hindi (Hi), Marathi (Mr), Odia (Od), Tamil (Ta), Telugu (Te), and Gujarati (Gu)). Among these languages, Hindi, Marathi, Odia and Gujarati are Indo-Aryan languages, while Tamil and Telugu fall under the category of Dravidian languages. As mentioned above, the authors considered six Indian languages ie., Hi, Mr, Od, Ta, Te and Gu, among these languages except Hi and Mr all others have different orthography (grapheme style). Moreover, every languages has its own training set they are (X_{Hi},L_{Hi}), (X_{Mr},L_{Mr}), (X_{Od},L_{Od}), (X_{Ta},L_{Ta}), (X_{Te},L_{Te}), and (X_{Gu},L_{Gu}), where X corresponds to the input acoustic sequence and L represents the label for target acoustic sequence. Initially, the monolingual systems are built for each language and trained with corresponding pairs which is acoustic sequence and its labels. Later, the authors attempt to show that the developed multilingual systems offer improved performance. As a part of multilingual system a joint acoustic model is trained, for which training dataset is constructed by pooling data from all the six languages, i.e., $(X_{all}, L_{all}) = (X_{Hi}, L_{Hi}) \cup (X_{Mr}, L_{Mr}) \cup (X_{Od}, L_{Od}) \cup (X_{Ta}, L_{Ta}) \cup (X_{Te}, L_{Te}) \cup (X_{Gu}, L_{Gu})$. The joint acoustic model is a single acoustic where the parameters are shared across all the six languages. This keeps the authors on similar lines with other research done on the development of ASR systems for low resource languages. In this work, a joint acoustic model-based ASR has been developed using different acoustic models. A joint acoustic model (JAM), which recognizes multiple languages with a single acoustic model, is widely appreciated. Many research groups have been actively working on this for the past few years (Chen et al., 2014). The usage of the joint acoustic model leverages the cross-lingual knowledge transfer. It reduces the complexity in the ASR pipeline significantly compared with the monolingual model as it has to maintain one model per language. The results show that the multilingual TDNN system result in lower word error rates (WER). We use KenLM (Heafield, 2011) and Recurrent Neural Network (RNN) based language models (Mikolov et al., 2010a) for decoding and rescoring respectively.

The organization of the remainder of the paper is as follows: Section 2 details the proposed lexicon along with the rationale behind the chosen mappings and database analysis. Section 3 explains the experimental setup. Section 4 discuss experimental results and few of the analysis which we have carried on. Finally, the study is concluded in Section 5, with possible future research directions to explore.

2 Dataset & Lexicon Details

INTERSPEECH 2018 has organized ASR Challenge as a special session (Srivastava et al., 2018). As part of the challenge, 40 hours of speech data has been released for three languages of Gujarati, Telugu, and Tamil. In continuation, INTERSPEECH 2021 has extended the challenge to six languages, i.e., Hindi (Hin), Marathi (Mar), Odia (Odi), Tamil (Tam), Telugu (Tel) and Gujarati (Guj) respectively (Diwan et al., 2021). The speech data statistics for the six languages are tabulated in the Table 1. To maintain a uniform sampling rate across all the languages, the authors have downsampled all the wave files to 8kHz while building the multilingual system.

Table 1: Database statistics for training and test sets (where Trn and Tst indicates Training and Testing sets respectively)

	Hindi		**Marathi**		**Odia**		**Tamil**		**Telugu**		**Gujarati**	
	Trn	**Tst**	**Trn**	**Tst**	**Trn**	**Tst**	**Trn**	**Tst**	**Trn**	**Tst**	**Trn**	**Tst**
fs (KHz)	8	8	8	8	8	8	16	16	16	16	16	16
# Hours	95.05	5.55	93.89	5.0	94.54	5.49	40	5	40	5	40	5
# Utt	99926	3843	79432	4675	59782	3471	39131	3081	44882	3040	22807	3075
Avg dur	5.2	6.0	4.5	5.8	3.9	5.2	3.6	5.8	3.2	5.9	6.3	5.8

2.1 Database Analysis

From Table 1, it is observed that Gujarati has the least number of utterances when compared to other languages in the database. Hindi has the maximum number of utterances when compared to others. Among the six languages, Hindi and Marathi follow the same orthography. So, there is a chance that either of the languages might get benefited while training together. The text data statistics for all the six languages are shown in Table 2.

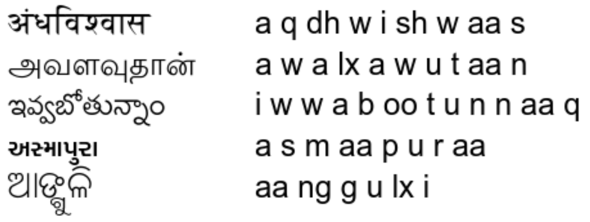
The last row of the table corresponds to number of graphemes. Among the total number of graphemes mentioned, for each language, the number of diacritic marks[2] are 16,16,16,7,17,17 respectively for Hindi, Marathi, Odia, Tamil, Telugu, and Gujarati. In analysis of the text data the authors found that, for a given word, the transcriptions are different in some cases. There are very few proper nouns in the database. In some utterances it is found that even English words are present but in the transliterated form. For example, *copy* $\rightarrow$ कॉपी, *book*$\rightarrow$ बुक. So this paper tries to capture such variations by exploring different models to empirically see which performs better in the present scenario. Throughout this paper, the authors use multilingual approaches to solve the data scarcity problem.

2.2 Lexicon

In building a low-resource speech recognition system, the lexicon plays a vital role in providing a phonetic representation for a given word sequence. In this section, the authors will be discussing about the unified parser which was introduced by (Baby et al., 2016). The proposed parser was primarily used for speech synthesis task across all the languages. The parser has two-folds:

- In the first fold, for a given UTF-8 word sequence it converts into corresponding syllables. Recently this type of approach has been explored by (Shetty and Umesh, 2021) for building end-end speech recognition system to improve the performance of low-resource Indian Languages.
- Later, in the second fold, letter-to-sound rules are applied and the corresponding phone sequence is generated. This type of parser is popular in building Indic TTS. The authors hypothesize this will work in their ASR pipeline. Therefore, in this paper, a similar approach is followed to build low-resource speech recognition system for Indian languages.

We have come up with a unique parser to generate the pronunciation sequence for all the words as shown in Figure 1.



अंधविश्वास	a q dh w i sh w aa s
அவளவுதான்	a w a lx a w u t aa n
ఇవ్వబోతున్నాం	i w w a b oo t u n n aa q
અસ્માપુરા	a s m aa p u r aa
ଆଙ୍ଗୁଳି	aa ng g u lx i

Figure 1: A example of Common Label Set based lexicon generated for Indian Languages

3 Speech Recognition Experimental Setup

In this section, the authors describe the experimental setup for both the monolingual and multilingual ASR systems. The word error rate (WER) is the metric used to evaluate the performance of ASR systems throughout this paper.

3.1 Language Modeling

The language model (LM) tries to estimate the probability of a hypothesized word sequence by learning the words from the text corpora. A Kneser-Ney (Chelba et al., 2010) trigram LM is built using SRILM toolkit (Stolcke, 2002) by normalizing training corpus. Recurrent neural networks

[2]A diacritical mark is a symbol that tells a reader how to pronounce a letter.

Table 2: Lexical analysis for training and test sets (where Trn and Tst indicates Training and Testing sets respectively)

	Hindi		Marathi		Odia		Tamil		Telugu		Gujarati	
	Trn	Tst	Trn	Tst	Trn	Tst	Trn	Tst	Trn	Tst	Trn	Tst
Uniq sent	4506	386	2543	200	820	65	30329	3060	34176	2997	20257	3069
Uniq words	6092	1681	3245	547	1584	334	50124	12279	43270	10859	39428	10482
OOV (%)	26.17		25.59		17.91		33.19		28.82		17.02	
# graphemes	69		61		68		50		64		65	

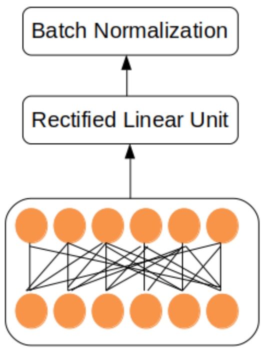
(RNN) based LM helps in preserving the information due to its feedback mechanism in its architecture (Mikolov et al., 2010b). It tries to take the previous information, $h_i = w_{i-1}, ..., w_1$ to predict the current word in sequence w_i. RNNLM has an input layer consisting of history vector h_i, previous word vector w_{i-1} and v_{i-2} is the context vector. The activation function used in this RNNLM is softmax. The input and output layer calculate the RNNLM probabilities $P_{RNNLM}\ (w_i|w_{i-1}, v_{i-2})$ using this activation function. Similarly, this process is repeated for calculating the probability of the next word.

In this paper, we use a TDNN-LSTM system for bulding RNNLM. The TDNN-LSTM has three LSTM layers with 1024 cells, 256 dimension projection and 9 layers of 1024 neurons. The L^2-regularization for hidden layers is 0.01 and output softmax is 0.004.

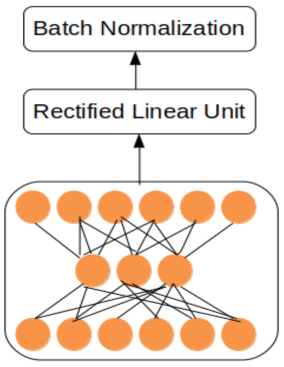
3.2 Acoustic Modeling

Sequence-trained TDNN architecture (Peddinti et al., 2015) is explored for building the baseline acoustic model using Kaldi toolkit (Povey et al., 2011). For training the baseline, the alignments from Gaussian Mixture Model-Hidden Markov Model (GMM-HMM) are considered. 13-dimensional Mel-Frequency Cepstral Coefficients (MFCC) features are used for mono-phone training. Next, Δ and $\Delta\Delta$ features are used for tri-phone modeling. The alignments generated from speaker adaptive training (SAT) based tri-phone models are used in the training of baseline TDNN. Feature space Maximum Likelihood Linear Regression (FMLLR)(Yao et al., 2012) is opted for SAT with 6000 tied states.

The input features to TDNN were 40 dimensional high resolution MFCCs with 100 dimensional iVectors for speaker adaptation (Madikeri et al., 2016). Initially, a three way speed perturbation is performed on the training data with 0.9, 1.0 and 1.1. Later, a volume perturbation has been



(a) Feed-forward layer in conventional TDNN system.



(b) Factorized layer in low rank TDNN system

Figure 2: Difference between a normal TDNN and low rank TDNN

adapted with a random factor between 0.9 to 2. All these perturbations are extracted for training iVectors.

Low Rank TDNN based architecture is explored which is different from the conventional TDNN. In this low rank TDNN, a bottleneck linear layer after every affine transformation of batch normalised ReLU is applied with skip connections. As it is factorized at every linear layer pair of each ReLU unit, it is also referred as low rank TDNN. The low rank TDNN based recipe can be seen in Kaldi [3]. The difference between a normal TDNN layer and a low rank TDNN layer is shown in Figures 2a and 2b.

In the low rank TDNN, the dimension for the linear bottleneck is 256. In general, the skip connection takes the inputs and outputs of the previous layer and selects other prior layers that are appended to the previous ones. In this experimental setup, apart from the output, it receives three non-consecutive layers as a skip connection. For example, consider 1280 x (256 x 2) as a dimension of conventional TDNN; after considering three skip

[3] egs/swbd/s5c/local/chain/tuning/run_tdnn_7n.sh

Table 3: The WER(%) results of monolingual (mono), multilingual (multi) models. The results showing the impact of the language model (LM) with and without external text data, and different acoustic models are also reported.

Type	Model	Hindi	Marathi	Odia	Tamil	Telugu	Gujarati	Average
Mono	GMM-HMM	69.03	33.22	55.78	48.81	47.27	28.33	**46.88**
	SGMM	61.01	26.41	51.36	39.68	28.08	28.61	**40.85**
	TDNN	30.16	19.65	35.58	21.89	21.67	17.35	**24.80**
	Low Rank TDNN_11L + External text data	16.30	12.35	13.48	15.36	11.36	13.65	**13.73**
Multi	SGMM	38.69	30.6	39.68	36.96	35.68	33.65	**35.87**
	TDNN_8L	35.16	31.58	37.6	35.98	34.89	31.62	**34.48**
	TDNN_LSTM	26.58	15.62	29.68	20.12	20.02	17.89	**21.65**
	TDNN_BLSTM	36.47	15.14	28.89	19.63	20.1	17.02	**22.87**
	TDNN_13L	16.36	12.65	23.22	19.04	19.34	16.88	**17.91**
	Low Rank TDNN_11L	17.27	10.69	21.65	18.34	18.68	15.08	**16.95**
	Low Rank TDNN_11L +External Telugu text data	17.27	10.69	21.65	18.34	9.86	15.08	**15.48**
	Low Rank TDNN_11L +External 6 language text data	8.37	6.35	10.78	9.15	9.86	7.25	**8.62**

connections, the dimension would be 1280 x (256 x 5).

4 Experimental Results & Analysis

The experiment results for the six languages are reported in the Table 3. The monolingual results show that TDNN system with 8 layers and six million parameters performs better than SGMM and GMM-HMM based models for monolingual systems. The TDNN system consistently improves the WER performance for all languages. The best WER is obtained for Gujarati which is 17.35%. We hypothesize that is due to low OOV%.

A multilingual neural network was trained by pooling the six languages using common label set. The JAM was trained with a similar configuration as the low rank TDNN system described in the Section 4. The results are reported for the SGMM, 7-layer TDNN, TDNN-LSTM, TDNN-BLSTM, 8-layer TDNN, 13-layer TDNN and low rank TDNN respectively. The best performance is observed for the case of low rank TDNN. The performance of models trained using SGMM is poorer than the performance of TDNN based models.

The triphones which are modeled by GMM-HMM do not share the common distribution among the six languages. This has led to the poor performance when the JAM is trained using SGMM. JAM trained using low rank TDNN has yielded better performance than TDNN and SGMM based systems. Unlike SGMMs, low rank TDNN and TDNN acoustic models are trained to model long-term temporal variations. The variabilities caused due to the presence of six languages have been effectively handled using low rank TDNNs due to the attributes of the architecture like skip connection, bottleneck linear transformation layer and batch normalization. From the last row of the Table 3, it is evident that due to inclusion of skip connections in low rank TDNN (with and without external text data), the performance of multilingual ASR system across all the languages has improved. Initially external text is collected for Telugu language and RNNLM is built using this. The authors evaluate this on low rank TDNN acoustic model. The collected external text and lexicon created using this can be found here [4]. The multilingual system

[4]External crawled data can be found here https://github.com/mirishkarganesh/icon_submission

with the same configurations as before was built with added external Telugu text. Due to which, as mentioned in the Table 3, the performance for Telugu language has improved. Motivated by the improvement in performance of ASR for Telugu, this was extended to all languages. The external text for other five languages is taken from AI4Bharat[5]. Similar improvement is observed for all languages with added external text.

5 Conclusion & Future Work

In this paper, a TDNN based multilingual ASR system for six Indian languages, i.e., Hindi, Marathi, Odia, Tamil, Telugu, and Gujarati was explored. Our experiment results show that the multilingual models achieve comparable results to the monolingual models when the parameters are in a comparable range. In few cases, ASR performance improved by including external text data while building languages model. CLS is investigated for studying the effectiveness of JAM for building a multilingual ASR system for six Indian languages. It is observed that low rank TDNN has shown superior performance over conventional TDNNs. Since most Indian languages are syllabic in nature and share a common phonetic space, the authors believe that the CLS approach can be further extended to more Indian languages in future. The feasibility of adapting LM along with the AM can also be explored for improving the performance of low resource multilingual ASR system. As India is a multilingual society, it is common occurrence for code-switching to be observed. In general, the authors would like to focus on two types of code-switching; (i) intra-sentential and (ii) inter-sentential. In the regular conversation, people try to switch, that is, the language exchange takes place at the sentence boundaries, and in the latter case, the languages switch into sentences, thus, creating a more complex problem. The six language multilingual model which we have built can deal with the inter-sentential cases. However, in Hindi and Marathi both intra and inter-sentential cases works as these two languages share same grapheme structure. Therefore, our future work will focus on recognizing intra-sentential code-switching utterances by exploring different architecture and utilizing monolingual data. The experimental findings from this paper will benefit researchers planning to build multilingual ASR systems for syllabic languages. We hope our work will encourage future research that leverages the findings.

[5]AI4Bharat text data can be found here `https://indicnlp.ai4bharat.org/corpora/`

References

Arun Baby, NL Nishanthi, Anju Leela Thomas, and Hema A Murthy. 2016. A unified parser for developing indian language text to speech synthesizers. In *International Conference on Text, Speech, and Dialogue*, pages 514–521. Springer.

Ciprian Chelba, Thorsten Brants, Will Neveitt, and Peng Xu. 2010. Study on interaction between entropy pruning and kneser-ney smoothing. In *Eleventh Annual Conference of the International Speech Communication Association*.

Dongpeng Chen, Brian Mak, Cheung-Chi Leung, and Sunil Sivadas. 2014. Joint acoustic modeling of triphones and trigraphemes by multi-task learning deep neural networks for low-resource speech recognition. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5592–5596. IEEE.

Jaejin Cho, Murali Karthick Baskar, Ruizhi Li, Matthew Wiesner, Sri Harish Mallidi, Nelson Yalta, Martin Karafiat, Shinji Watanabe, and Takaaki Hori. 2018. Multilingual sequence-to-sequence speech recognition: architecture, transfer learning, and language modeling. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 521–527. IEEE.

Ekapol Chuangsuwanich. 2016. Multilingual techniques for low resource automatic speech recognition. Technical report, Massachusetts Institute of Technology Cambridge United States.

Jia Cui, Brian Kingsbury, Bhuvana Ramabhadran, Abhinav Sethy, Kartik Audhkhasi, Xiaodong Cui, Ellen Kislal, Lidia Mangu, Markus Nussbaum-Thom, Michael Picheny, et al. 2015. Multilingual representations for low resource speech recognition and keyword search. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 259–266. IEEE.

Amit Das and Mark Hasegawa-Johnson. 2015. Cross-lingual transfer learning during supervised training in low resource scenarios. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Anuj Diwan, Rakesh Vaideeswaran, Sanket Shah, Ankita Singh, Srinivasa Raghavan, Shreya Khare, Vinit Unni, Saurabh Vyas, Akash Rajpuria, Chiranjeevi Yarra, Ashish Mittal, Prasanta Kumar Ghosh, Preethi Jyothi, Kalika Bali, Vivek Seshadri, Sunayana Sitaram, Samarth Bharadwaj, Jai Nanavati, Raoul Nanavati, Karthik Sankaranarayanan, Tejaswi Seeram, and Basil Abraham. 2021. Multilingual and code-switching asr challenges for low resource indian languages. *arXiv preprint arXiv:2104.00235*.

Thierry Dutoit. 1997. *An introduction to text-to-speech synthesis*, volume 3. Springer Science & Business Media.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the sixth workshop on statistical machine translation*, pages 187–197.

Aizhan Imankulova, Raj Dabre, Atsushi Fujita, and Kenji Imamura. 2019. Exploiting out-of-domain parallel data through multilingual transfer learning for low-resource neural machine translation. *arXiv preprint arXiv:1907.03060*.

Chunxi Liu, Qiaochu Zhang, Xiaohui Zhang, Kritika Singh, Yatharth Saraf, and Geoffrey Zweig. 2019. Multilingual graphemic hybrid asr with massive data augmentation. *arXiv preprint arXiv:1909.06522*.

Liang Lu, Arnab Ghoshal, and Steve Renals. 2013. Acoustic data-driven pronunciation lexicon for large vocabulary speech recognition. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 374–379. IEEE.

Srikanth Madikeri, Subhadeep Dey, Petr Motlicek, and Marc Ferras. 2016. Implementation of the standard i-vector system for the kaldi speech recognition toolkit. Technical report, Idiap.

Yajie Miao, Florian Metze, and Shourabh Rawat. 2013. Deep maxout networks for low-resource speech recognition. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 398–403. IEEE.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010a. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010b. Recurrent neural network based language model. volume 2, pages 1045–1048.

Ayushi Pandey, Brij Mohan Lai Srivastava, and Suryakanth V Gangashetty. 2017. Adapting monolingual resources for code-mixed hindi-english speech recognition. In *2017 International Conference on Asian Language Processing (IALP)*, pages 218–221. IEEE.

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*.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. 2011. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, CONF. IEEE Signal Processing Society.

Kishore Prahallad, E Naresh Kumar, Venkatesh Keri, S Rajendran, and Alan W Black. 2012. The iiit-h indic speech databases. In *Thirteenth annual conference of the international speech communication association*.

Afshin Rahimi, Yuan Li, and Trevor Cohn. 2019. Multilingual ner transfer for low-resource languages.

B Ramani, S Lilly Christina, G Anushiya Rachel, V Sherlin Solomi, Mahesh Kumar Nandwana, Anusha Prakash, S Aswin Shanmugam, Raghava Krishnan, S Kishore Prahalad, K Samudravijaya, et al. 2013. A common attribute based unified hts framework for speech synthesis in indian languages. In *Eighth ISCA Workshop on Speech Synthesis*.

D Raj Reddy. 1976. Speech recognition by machine: A review. *Proceedings of the IEEE*, 64(4):501–531.

Vishwas M. Shetty and S. Umesh. 2021. Exploring the use of Common Label Set to Improve Speech Recognition of Low Resource Indian Languages. pages 7228–7232.

Brij Mohan Lal Srivastava, Sunayana Sitaram, Rupesh Kumar Mehta, Krishna Doss Mohan, Pallavi Matani, Sandeepkumar Satpal, Kalika Bali, Radhakrishnan Srikanth, and Niranjan Nayak. 2018. Interspeech 2018 low resource automatic speech recognition challenge for indian languages. In *SLTU*, pages 11–14.

Andreas Stolcke. 2002. Srilm-an extensible language modeling toolkit. In *Seventh international conference on spoken language processing*.

Masahide Sugiyama, Hidehumi Sawai, and Alexander H Waibel. 1991. Review of tdnn (time delay neural network) architectures for speech recognition. In *1991., IEEE International Sympoisum on Circuits and Systems*, pages 582–585. IEEE.

Pawel Swietojanski, Arnab Ghoshal, and Steve Renals. 2012. Unsupervised cross-lingual knowledge transfer in dnn-based lvcsr. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 246–251. IEEE.

Samuel Thomas, Kartik Audhkhasi, and Brian Kingsbury. 2020. Transliteration based data augmentation for training multilingual asr acoustic models in low resource settings. *Proc. Interspeech 2020*, pages 4736–4740.

Samuel Thomas, Sriram Ganapathy, and Hynek Hermansky. 2012. Multilingual mlp features for low-resource lvcsr systems. In *2012 IEEE International*

Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4269–4272. IEEE.

Hari Krishna Vydana, Krishna Gurugubelli, Vishnu Vidyadhara Raju Vegesna, and Anil Kumar Vuppala. 2018. An exploration towards joint acoustic modeling for indian languages: Iiit-h submission for low resource speech recognition challenge for indian languages, interspeech 2018. In *INTERSPEECH*, pages 3192–3196.

Wikipedia. 2021. List of languages by number of native speakers in India. https://en.wikipedia.org/w/index.php?title=List_of_languages_by_number_of_native_speakers_in_India&oldid=1012063187. [Online; accessed 22-March-2021].

Haihua Xu, Van Hai Do, Xiong Xiao, and Eng Siong Chng. 2015. A comparative study of bnf and dnn multilingual training on cross-lingual low-resource speech recognition. In *Sixteenth annual conference of the international speech communication association*.

Kaisheng Yao, Dong Yu, Frank Seide, Hang Su, Li Deng, and Yifan Gong. 2012. Adaptation of context-dependent deep neural networks for automatic speech recognition. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 366–369. IEEE.

Improving Sinhala Speech Recognition Through e2e LF-MMI Model

Buddhi Gamage, Randil Pushpananda, Thilini Nadungodage, Ruvan Weerasinghe
Language Technology Research Laboratory (LTRL)
University of Colombo School of Computing, Sri Lanka
{bud, rpn, hnd, arw}@ucsc.cmb.ac.lk

Abstract

Automatic speech recognition (ASR) has experienced several paradigm shifts over the years from template-based approaches and statistical modeling to the popular GMM-HMM approach and then to deep learning hybrid model DNN-HMM. The latest shift is to end-to-end (e2e) DNN architecture. We present a study to build an e2e ASR system using state-of-the-art deep learning models to verify the applicability of e2e ASR models for the highly inflected and yet low-resource Sinhala language. We experimented on e2e Lattice-Free Maximum Mutual Information (e2e LF-MMI) model with the baseline statistical models with 40 hours of training data to evaluate. We used the same corpus for creating language models and lexicon in our previous study, which resulted in the best accuracy for the Sinhala language. We were able to achieve a Word-error-rate (WER) of 28.55% for Sinhala, only slightly worse than the existing best hybrid model. Our model, however, is more context-independent and faster for Sinhala speech recognition and so more suitable for general purpose speech-to-text translation.

1 Introduction

There are two main architectures in training the ASR system. They are Statistical ASR architecture and End-to-End(e2e) Deep Neural architecture. Statistical ASR was the state of the art for many years, however after the year 2015, researchers tend to move towards e2e ASR systems due to the higher results. The main difference between these two architectures is the number of models needed to create, in training the ASR system. In Statistical ASR, it needs 3 types of models and they are acoustic models, pronunciation models and the language model. But in e2e ASR, it will compress those 3 models into a single Deep-Neural-Network (DNN) (Wang et al., 2019).

So many research have been done using e2e architecture for creating ASR systems for different languages since this is a new trending area of Natural-Language-Processing (NLP) and speech recognition. And there are previous researches conducted for English speech recognition that show better results when using e2e architecture than the traditional statistical approach (Park et al., 2019).

Currently, there are previous and ongoing researches on building ASR systems for Sinhala language using statistical ASR architecture, Gaussian mixture model with Hidden Markov model (GMM-HMM) based models and Hybrid Deep-Neural-Network with Hidden Markov model (DNN-HMM) based models (Gamage et al., 2020; Karunathilaka et al., 2020). E2e architecture would be a new approach for Sinhala ASR and it will help to improve the available resources as well. Especially, e2e architecture opens the doorway to transfer learning, which is the new trending for low resource speech recognition, to improve the accuracy. These domains for speech recognition were popular in later 2019 among the researchers and it is essential to have models trained in e2e architecture in Sinhala language to get a generic idea when accessing these domains (Stoian et al., 2020).

In this paper, we present a study on e2e DNN architecture based ASR systems for Sinhala speech recognition using e2e LF-MMI model. The performances of the e2e model will be evaluated and compared with the statistical models such as GMM-HMM, DNN-HMM and combinational models of SGMM-DNN.

The paper is organized as follows. Section 2 presents the related studies, Section 3 describes the methodology, data preparation and implementation in greater detail. Section 4 describes the results and evaluation. Section 5 presents the conclusions and the future work.

2 Literature review

More than 30 years later also this methodology still predominates in ASR. Nowadays, most practical speech recognition systems are based on the statistical approach (Wang et al., 2019).

With the improvement of deep learning, DNN is introduced for creating acoustic models. The role of DNN is to calculate the posterior probability of the HMM state, by replacing the traditional GMM observation probability. So DNN-HMM models become the state-of-the-art ASR model by achieving better results than GMM-HMM models (Wang et al., 2019). The training process and decoding process of the HMM-based model determines whether it faces the following difficulties in actual use.

- The training process is very complicated and it is difficult to perform global optimization.
- When constructing HMM based models, they made an assumption that the 3 models are independent from each other. This simplify the model creation but this is not an actual match (Wang et al., 2019).

Due to the above-mentioned shortcomings or anomalies in the HMM-based models, more research was carried out in the e2e architecture with the trending of deep learning. The end-to-end model is a system that directly maps input audio sequence to sequence of words or other graphemes (Rao et al., 2017). So direct mapping of utterances to character sequence is conducted where no intermediate states like calculating posteriors in the output (Wang et al., 2019).

Data alignment is the major problem in both HMM based and e2e models but e2e models require soft alignments where HMM based models use forced alignments. However, main problem in e2e architecture is that it requires a large amount of speech data to achieve higher accuracy in recognition (Wang et al., 2019). Till year 2018 low resource speech recognition systems are never used e2e architecture because this architecture is used in Large Vocabulary Continuous Speech Recognitio (LVCSR). (Povey et al., 2016) paper shows that, abundance of training data make the system lagged comparable to hybrid DNN systems when trained on smaller training sets.

Interspeech 2018 Low Resource Automatic Speech Recognition Challenge for Indian Languages research conducted by Microsoft Corporation in India created a challenge by giving 50 hours of transcribed speech in each Tamil, Telugu and Gujarati which are three main indian languages and asked participants to build ASR systems restricted to this dataset. Evaluation carried out on a blind test set (Srivastava et al., 2018). The ISI-Billa system presented to address the above challenge (Billa, 2018) was an EESEN based end-to-end multilingual LSTM network trained using the Connectionist Temporal Classification (CTC) training criterion . This is the first time to use an e2e model for south asian languages. Both monolingual and multilingual systems trained using this model and it outperformed the baseline models in all three languages (Srivastava et al., 2018).

After year 2019, transfer learning and unsupervised learning techniques have become famous in the speech recognition domain. One of the major improvements was to tackle the low resource problems researchers are introducing in these domains (Bataev et al., 2018). In the Investigation of transfer learning for ASR using LF-MMI trained neural networks paper they used weight transfer and multitask learning transfer learning techniques to address the low resource problem (Ghahremani et al., 2017). Meta Learning For End-to-end Low-resource Speech Recognition paper shows that using multilingual CTC models can be used to improve the accuracy of the ASR using Meta learning techniques (Hsu et al., 2020). In wav2vec: Unsupervised Pre-training for Speech Recognition paper used a model is trained using the Auto Segmentation Criterion. Untranscribed web audio for low resource speech recognition paper has introduced semi-supervised training is done by using lattice-free maximum mutual information (LF-MMI) to untranscribed data (Carmantini et al., 2019).

3 Approach

3.1 Data Preparation

Data preparation is the most important task in the ASR pipeline because to have a reliable ASR, it highly depends on the consistency and integrity of the data preparation step (Gamage et al., 2020). We used Kaldi toolkit in the study and data preparation is done accordingly. Since the training scripts of e2e LF-MMI models does not allow segment file which has the length of each utterance in a single audio file, we had to split the audio recored in praat to have a single audio wave file to a single utterance in training models.

3.1.1 Dataset

Here we have used the collected recordings from Language Technology Research Laboratory (LTRL) of University of Colombo School of Computing (UCSC) which has 40 hours of training data which have been gathered using Praat and Redstart tools. Training the models involves a total data set from 113 native speakers where 79 are female, and 34 are males. The training data set has audio recordings from 67 females and 27 males speakers, and the total utterances are 17848 sentences, which is 25h of speech data. As the validation data set,2002 speech utterances from 8 females and 3 male speakers are taken for fine-tuning the models. Testing the models involve a data set from 4 female speakers and 4 male speakers where they utter 80 speech sentences altogether. Training has done in 16kHz sample rate and refer (Gamage et al., 2020) for more details. The overall details about the data sets are given in table 1.

Dataset	Male	Female	Utterances
Train	27	67	17848
Dev	3	8	2002
Test	4	4	80

Table 1: Details of train,validation and test data sets

3.1.2 Lexicon

Lexicon has the mapping of words with the relevant spoken phone sequences and it is a major part of the pronunciation model in a statistical ASR system (Gamage et al., 2020). For creating lexicon, ”Sinhala G2P Conversion” (Nadungodage et al., 2018) and ”Subasa Transliterator” tools are used and please refer (Gamage et al., 2020) for more details.

3.1.3 Corpus

We used 3 corpora namely, UCSC Novel Corpus (90000 unique sentences), Chatbot Corpus (388 unique sentences) and the corpus created using active learning method (20000 unique sentences) to create the corpus for the study. By combining the above corpora, a new corpus is created to generate ngram language models. Summary of the corpus is available in table 2. SRILM (Stolcke, 2002) toolkit and KenLM (Heafield, 2011) toolkit are used to create the n-gram language models. After calculating perplexities for testing dataset we selected a 4-gram language model through the study. Details of the Language Models are represented in table 3.

Vocabulary Size	243339
Total number of Sentences	119621
Total number of words	1194940

Table 2: Corpus Statistics

Language Model	Perplexity
Witten-Bell 3grams	9.393376
Witten-Bell 4grams	8.108833

Table 3: Perplexities of created Language Models

3.2 Baseline models

Mainly, 4 baseline models are considered in the study excluding monophone and triphone models as done for Sinhala language mentioned in (Gamage et al., 2020). Those baseline models are,

- Subspace Gaussian mixture model with MMI (SGMM+MMI)
- Hybrid System (Dan’s DNN)
- Hybrid System (Keral’s DNN)
- Combination SGMM + Dan’s DNN

Detailed descriptions of creating baseline models are presented in (Gamage et al., 2020).Usage of Combinational Acoustic

Models(DNN-HMM and SGMM) and Identifying the Impact of Language Models in Sinhala Speech Recognition (Gamage et al., 2020) paper uses 30 hours of training data. Additionally we have used 10 more hours with the same dataset.

Mel Frequency Cepstral Coefficient (MFCC) feature extraction is done using 13 MFCC with downsampling and zero order coefficient as it is the standard measurement, and features are extracted every 10ms with the 25ms Hamming window (Povey et al., 2011). Followed by, we trained models with monophone training, triphone training with delta feature computation, Linear Discriminant Analysis (LDA) with Maximum Likelihood Linear Transform(MLLT) and Speaker Adaptive Training (SAT). Alignments of LDA+MLLT+SAT model are used to train SGMM+MMI models and Hybrid models. DNN models are influenced by Keral's recipe and Dan's recipe mentioned in Kadli (Povey et al., 2011). Parameters for above models are mentioned in (Gamage et al., 2020), in detail. Results obtained in baseline models are represented in table 4.

3.3 Proposed E2e Lattice-Free Maximum Mutual Information (LF-MMI) model

We used the default Neural Network (NN) architecture to train WSJ dataset mentioned in training recipes. We used Factored Time Delay Neural Networks (TDNNf) according to the standard Kaldi WSJ recipe. This neural network has 13 TDNNf layers and a rank reduction layer. The number of units in the TDDNf consists of 1024 and 128 bottleneck units. The default hyperparameters of the standard recipe were used with the number of epochs 10, 30 and 50. (Hadian et al., 2018).

We chose the phone based training to create the e2e models. Architecture used to create e2e LF-MMI models are represented in figure 1.

Unlike in baseline models, 40-dimensional MFCC features are extracted from 25ms frames every 10ms because it is the default used in WSJ recipe (Hadian et al., 2018). Zero mean and unit variance normalization techniques are used per-speaker basis and no other feature normalization or feature transform is used. Unlike in baseline models, we do not perform re-alignments during the training here.

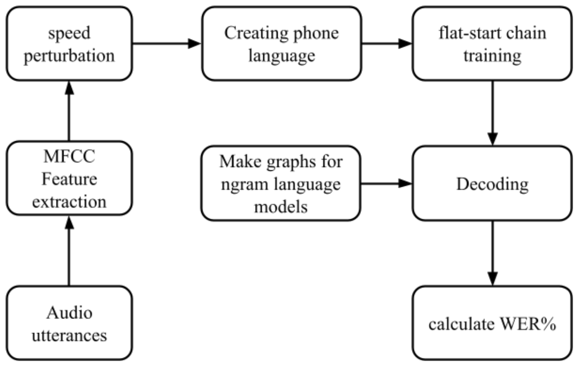


Figure 1: e2e LF-MMI model Architecture

Data is augmented with 2-fold speed perturbation in all the experiments because it modifies the length of each utterance to the nearest of the distinct lengths (Hadian et al., 2018). Otherwise, we can pad each utterance with silence to reach one of the distinct lengths.

Unlike in statistical ASR , e2e ASR decodes the utterance into character sequence. So we need a phone language for the denominator graph (Povey et al., 2016) to decode utterances. Then we start training the models in Kaldi toolkit with NN settings mentioned above. A different lang directory which contains the information of n-gram language models, can be used with a wordlist and language model of our choice to train the models, as long as phones.txt is compatible. The mkgraph.sh script helps to train the e2e models using such language models.

4 Results and Evaluation

The server used for training of all deep neural architectures and the decoding of the models contains 4 GPUs - GeForce RTX 2080 Ti of 10.8GB each. All 4 GPUs have been used when training, Thereby accelerating the deep learning training process by leveraging CUDA. The performance of e2e Sinhala ASR systems are evaluated in terms of accuracy on the recordings taken with the noise environment. This accuracy can be obtained by calculating either Word Error Rate (WER) or Sentence Error Rate (SER). WER is the number of words that are wrongly identified out of the total number of words in the audio sample used for recognition. SER is the number of sentences that are improperly identified out of the total number of sentences (Karunathilaka et al., 2020).The WER is widely used to discrete and compare speech recognition systems and we also used the WER throughout the study.

4.1 Baseline models

The combined SGMM+DNN statistical model is created with 40 hours of training data mentioned above. As discussed, the hybrid DNN model and SGMM+MMI models are created on top of the alignments of LDA+MMLT+SAT (tri3) triphone model. In (Gamage et al., 2020) statistical architecture achieved 31.72 %WER for only 30 hours of training data in the combined model. Table 5 represents the comparison of the results with the same architecture of previous study with 10hours more data.

Model	Test set (WERs)	Dev set (WERs)
mono	47.43	3.78
tri1	33.08	2.87
tri2	32.78	3.19
tri3	35.80	3.10
sgmm2_4	32.33	2.89
SGMM2 + MMI Training	31.72	2.87
Hybrid System (Dan's DNN)	27.79	2.29
Hybrid System (Keral's DNN)	27.95	3.96
Combination SGMM + Dan's DNN	29.00	2.44
Combination SGMM + Keral's DNN	28.40	2.95

Table 4: Results of Baseline models with 40 hours of data

We can clearly identify in table 4 that Hybrid Dan's DNN model achieved the best WER which is 27.79% and it outperformed the combined model by 0.61% less WER. With 30 hours of training data SGMM2+MMI model achieved 34.14% WER and within 40 hours it achieved 31.72% WER with the improvement of 2.42% less WER. And Hybrid Dan's model had 35.50% WER with 30 hours of training data but using 10 hours more data it achieved 27.79% WER with 5.71% improvement in WER. So we can clearly identify that with more data, hybrid DNN models performing well. This is a well known fact that DNN models have the higher accuracy in Speech recognition with the higher data available for training.

Model	WERs of	
	30 hour dataset	40 hour dataset
SGMM2 + MMI Training	34.14	31.72
Hybrid System (Dan's DNN)	35.50	27.79
Hybrid System (Keral's DNN)	36.33	27.95
Combination SGMM + Dan's DNN	31.72	29.00

Table 5: WER comparison of baseline models after increasing 10 more hours of auido data to the same dataset

4.2 E2e LF-MMI Models

Even though LF-MMI model is loosely coupled with HMM it can act as e2e model with monophones or full bi-phones by removing the context dependency tree alignments. In this study we use full left bi-phones so every possible pair of phones have a separate HMM model.

Table 6 represents the WERs achieved in e2e LF-MMI models and was able to get 28.55% WER with 10 epochs in Kaldi. When using GPU for training in Kaldi, it considers available 4 GPUs as a single GPU using exclusive mode and we can give higher frames when training. So for each epochs we used 3 million frames per iterations.

Ephocs	Test set (WERs)	Dev set (WERs)
10	28.55	2.27
30	32.18	2.90
50	33.27	2.02

Table 6: WER comparison of e2e LF-MMI models

4.3 Evaluation

Following 3 sentences are taken from 3 random persons. Recording of those utterances have been done in their own environment with their own equipment and they had 44.1 Hz sample rate. Hybrid Dan's model was taken as the baseline model because it had the lowest WER among other baseline models.They are compared with the accurate e2e LF-MMI model. Recorded utterances are fed into the above models and resulted outputs are represented in table 7 to 9.

Test sentence 1 in figure 7 is a news reading and (Gamage et al., 2020) paper shows

sentence	Russia's first humanoid robot arrives at International Space Station
Utterance	රුසියාව විසින් ප්‍රථම වරට හදුන්වාදුන් හියුමනොයිඩ් රොබෝවරයා ජාත්‍යන්තර අභ්‍යාවකාශ මධ්‍යස්ථානය වෙත ළහා වී තිබේ
Baseline	රුසියාව විසින් ප්‍රථමවරට හදුන්වා දුන්නේ මා නොයෙක් ඔයා බෝරියා ජාත්‍යන්තර අභ්‍යවකාශ මධ්‍යස්ථානය වෙත ළහා වී තිබේ
e2e LF-MMI	වාසිහු විසින් පැතුම් හා පහම හාද නො වීම නො යි ඔහු පෝමරය ජාත්‍යන්තර අභ්‍යවකාශ මධ්‍යස්ථානය වෙත ළහා වී තිබේ

Table 7: Analysis of test sentence 1

that the current Sinhala ASR system is well performed in the context of news readings and number readings. Because training transcriptions are mainly gathered in those areas. In this sentence also baseline model is well performed rather than other e2e models but "හියුමනොයිඩ් රොබෝවරයා" is not identified correctly. Those two words not being included in the lexicon and the corpus can be the reason. But in e2e LF-MMI model, we can see that it is trying to identify the word "හියුමනොයිඩ් " correctly by looking at "ම නො යි" "වීම නො යි".

sentence	The Cabinet also decided to provide President Maithripala Sirisena with his current official residence at Mahagama Sekara Mawatha, Colombo 7, after his retirement.
Utterance	ජනාධිපති මෛත්‍රීපාල සිරිසේන මහතා දැනට භාවිතාකරන කොළඹ හත මහගමසේකර මාවතේ පිහිටි නිලනිවස විශ්‍රාම ගැනීමෙන් පසුව ද ඔහුට ලබා දීමට කැබිනට් මණ්ඩලය තීරණය කරයි
Baseline	ඇස් ජනාධිපති මට පන්සල මහතා ඇයට භාවිතාකරන කොළඹ හත මහගමසේකර මාවතේ පිහිටි නිල නිවස විශ්‍රාම ගැනීමෙන් පසුව ද ඔහුට ලබා දීමට කැබිනට් මණ්ඩලය තීරණය කළහ
e2e LF-MMI	නාදපති මෛත්‍රීපාල සිරිසේන මහතා දැඩි භාවිතාකරන කොළඹ හත මහගමසේකර මාවතේ පිහිටි නිලනිවසට විශ්‍රාම ගැනීමෙන් පසුව ද ඔහුට ලබා දීමට කැවිලි බණ්ඩනය තීරණයක

Table 8: Analysis of test sentence 2

"ජනාධිපති මෛත්‍රීපාල සිරිසේන" and "කැබිනට් මණ්ඩලය" is not identified correctly in all models at to some extent. Those words are included in both lexicon and corpus but they are not identified correctly. Problem here is acoustic data is not enough to train, to have higher probability for those words because they are proper nouns. So n-gram language model dominated here to have higher probability in "ජනාධිපති මට" rather than "ජනාධිපති මෛත්‍රීපාල" because in previous study with less data those two words correctly identified in the baseline model. E2e model is trying to get the more accurate decoding. So we can identify that there is a misleading in Statistical models when using higher data. E2e models use character level decoding rather than word level decoding done in statistical architecture so that misleading is minimum in the e2e models and we can identify that in the above sentence.

sentence	I leave
Utterance	මම යනවා
Baseline	විවර
e2e LF-MMI	මම යනවා

Table 9: Analysis of test sentence 3

Test sentence 3 in figure 9 has a normal day to day talking accent. E2e LF-MMI model has correctly identify the utterance. Baseline model completely mis-identifies the utterance because it is context dependent when decoding and sentences with less number of words mostly have low accuracy because it uses word level decoding. Many other sentences had this observation so that we can identify that the e2e models are more context independent rather than statistical models even though the training data has a context dependency. To have a general ASR system, e2e techniques are more suitable.

5 Conclusion and Future Work

Even though there is a slightly better accuracy in statistical approach, using the e2e approach we created a less context dependent and faster model for Sinhala speech recognition for using general purpose speech-to-text transcription. We found out that using only statistical models (GMM+HMM, SGMM, SGMM+MMI) is not useful in further research conducted for Sinhala speech recognition that even hybrid system where DNN uses to calculate the posterior probabilities for the HMM perform far better than those traditional approaches.

Currently, we were able to achieve 28.55% WER for Sinhala e2e speech recognition using e2e LF-MMI implemented on Kaldi toolkit. This model also can be improved with fine tuning but this study is not going to fine tune and has used the basic implementations and recipes available for the WSJ dataset which have 80 hours of training data. Current domain of speech recognition moves toward addressing the low resource problem. There are large datasets available for English and France like languages with the state-of-the-art results. Common solution for addressing the low resource problem is to transfer learning from high resource language to a low resource language. In e2e LF-MMI technique transfer learning can be done by using weight transfer and multi-task training (Ghahremani et al., 2017). So from the results of this study it will be useful to do the necessary data augmentations and choosing parameters for transfer learning techniques mentioned above.

References

Vladimir Bataev, Maxim Korenevsky, Ivan Medennikov, and Alexander Zatvornitskiy. 2018. Exploring end-to-end techniques for low-resource speech recognition. In International Conference on Speech and Computer, pages 32–41. Springer.

Jayadev Billa. 2018. Isi asr system for the low resource speech recognition challenge for indian languages. In INTERSPEECH, pages 3207–3211.

Andrea Carmantini, Peter Bell, and Steve Renals. 2019. Untranscribed web audio for low resource speech recognition. In INTERSPEECH, pages 226–230.

Buddhi Gamage, Randil Pushpananda, Ruvan Weerasinghe, and Thilini Nadungodage. 2020. Usage of combinational acoustic models (dnn-hmm and sgmm) and identifying the impact of language models in sinhala speech recognition. In 2020 20th International Conference on Advances in ICT for Emerging Regions (ICTer), pages 17–22. IEEE.

Pegah Ghahremani, Vimal Manohar, Hossein Hadian, Daniel Povey, and Sanjeev Khudanpur. 2017. Investigation of transfer learning for asr using lf-mmi trained neural networks. In 2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pages 279–286. IEEE.

Hossein Hadian, Hossein Sameti, Daniel Povey, and Sanjeev Khudanpur. 2018. End-to-end speech recognition using lattice-free mmi. In Interspeech, pages 12–16.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In Proceedings of the sixth workshop on statistical machine translation, pages 187–197.

Jui-Yang Hsu, Yuan-Jui Chen, and Hung-yi Lee. 2020. Meta learning for end-to-end low-resource speech recognition. In ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 7844–7848. IEEE.

Hirunika Karunathilaka, Viraj Welgama, Thilini Nadungodage, and Ruvan Weerasinghe. 2020. Low-resource sinhala speech recognition using deep learning. In 2020 20th International Conference on Advances in ICT for Emerging Regions (ICTer), pages 196–201. IEEE.

Thilini Nadungodage, Chamila Liyanage, Amathri Prerera, Randil Pushpananda, and Ruvan Weerasinghe. 2018. Sinhala g2p conversion for speech processing. In SLTU, pages 112–116.

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. arXiv preprint arXiv:1904.08779.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. 2011. The kaldi speech recognition toolkit. In IEEE 2011 workshop on automatic speech recognition and understanding, CONF. IEEE Signal Processing Society.

Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur. 2016. Purely sequence-trained neural networks for asr based on lattice-free mmi. In Interspeech, pages 2751–2755.

Kanishka Rao, Haşim Sak, and Rohit Prabhavalkar. 2017. Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer. In 2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pages 193–199. IEEE.

Brij Mohan Lal Srivastava, Sunayana Sitaram, Rupesh Kumar Mehta, Krishna Doss Mohan, Pallavi Matani, Sandeepkumar Satpal, Kalika Bali, Radhakrishnan Srikanth, and Niranjan Nayak. 2018. Interspeech 2018 low resource automatic speech recognition challenge for indian languages. In SLTU, pages 11–14.

Mihaela C Stoian, Sameer Bansal, and Sharon Goldwater. 2020. Analyzing asr pretraining for low-resource speech-to-text translation. In ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 7909–7913. IEEE.

Andreas Stolcke. 2002. Srilm-an extensible language modeling toolkit. In Seventh international conference on spoken language processing.

Dong Wang, Xiaodong Wang, and Shaohe Lv. 2019. An overview of end-to-end automatic speech recognition. Symmetry, 11(8):1018.

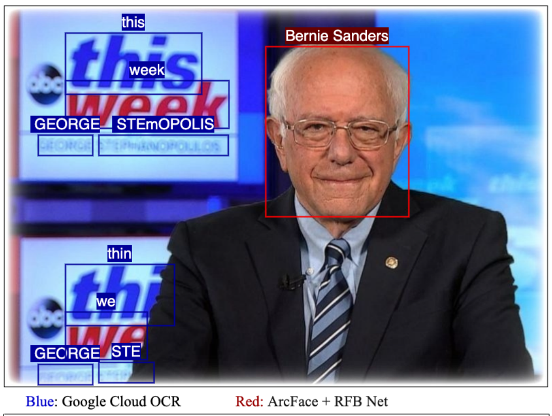
Towards Multimodal Vision-Language Models Generating Non-Generic Text

Wes Robbins
Montana State University
wesley.robbins10@gmail.com

Zanyar Zohourianshahzadi & Jugal Kalita
University of Colorado, Colorado Springs
{zzohouri,jkalita}@uccs.edu

Abstract

Vision-language models can assess visual context in an image and generate descriptive text. While the generated text may be accurate and syntactically correct, it is often overly general. To address this, recent work has used optical character recognition to supplement visual information with text extracted from an image. In this work, we contend that vision-language models can benefit from additional information that can be extracted from an image, but are not used by current models. We modify previous multimodal frameworks to accept relevant information from any number of auxiliary classifiers. In particular, we focus on person names as an additional set of tokens and create a novel image-caption dataset to facilitate captioning with person names. The dataset, Politicians and Athletes in Captions (PAC), consists of captioned images of well-known people in context. By fine-tuning pretrained models with this dataset, we demonstrate a model that can naturally integrate facial recognition tokens into generated text by training on limited data. For the PAC dataset, we provide a discussion on collection and baseline benchmark scores.



Our Model: bernie sanders standing in front of a screen that says "this week"	
Previous Model: a man giving a speech in front of a screen that says "ste"	

Figure 1: Our captioning model accepts tokens from several upstream classifiers, learns representations for tokens from different classifiers, and uses each token appropriately. By using the facial recognition token 'Bernie Sanders', our model's caption is more informative than previous work which just uses OCR.[1]

1 Introduction

Vision-language models combine deep learning techniques from computer vision and natural language processing to assimilate visual and textual understanding. Such models demonstrate visual and linguistic knowledge by performing tasks such as vision question answering (VQA) and image captioning. There are many applications of these tasks, including aiding the visually impaired by providing scene information and screen reading (Morris et al., 2018).

To perform a vision-language task, a model needs to understand visual context and natural language, and operate in a shared embedding space between the two. Approaches in the literature have improved performance by pre-training models for both visual context and language understanding (Chen et al., 2020; Lu et al., 2019; Su et al., 2019; Li et al., 2020; Tan and Bansal, 2019). These models have yielded accurate and semantically appropriate VQAs or captions. However, the text generated from these models are general and overlook content that allow for richer text generation with improved contextualization. For example, they ignore clearly visible text or the presence of well-known individuals.

To improve specificity in generated text, recent work has used optical character recognition (OCR)

[1]The previous model in Figure 1 is M4C Captioner (Sidorov et al., 2020) with weights from the M4C repository.

to incorporate text that appears in images (Zhu et al., 2021; Gao et al., 2020b; Mafla et al., 2021; Hu et al., 2020; Kant et al., 2020; Wang et al., 2021; Han et al., 2020; Liu et al., 2020; Yang et al., 2021). In many cases, this significantly enhances the usefulness of the generated text (Hu et al., 2020). Such frameworks include OCR as an additional input modality. This results in three modalities for VQA (image, question, and OCR) and two modalities for image captioning (image and OCR).

While using OCR allows enhancement of some generated text, specific information that exists in human-level description may also come from additional sources. Without proper nouns or other specific vocabulary, the generated text is at the risk of being awkwardly general, demonstrating a lack of shared knowledge that is expected in society. For example in Figure 1, arguably the most relevant content in the image is the presence of a well-known political figure. Consequently, a reasonable description of the image should include the name of the well-known figure, which is 'Bernie Sanders' is in this case, instead of generic "a man". This is notably absent in the caption from the previous model.

In this work, we propose the *special token approach*, a novel method for integrating tokens from several upstream vision classifiers into image captions.[2] We generalize the OCR input modality to accept additional helpful outputs from any number of auxiliary classifiers (Section 3.2). We use a rich feature representation for upstream tokens that allows the captioning model to learn to differentiate tokens from different classifiers (Section 3.3).

This method potentially allows a model to leverage easily available sophisticated libraries to recognize faces, scene-text, cityscapes, animal species, etc. We refer to all tokens from upstream sources, including OCR tokens, as special tokens. In this work, we focus on using person names and scene-text as example special tokens.

To facilitate using person names in image captions, we create a novel image-caption dataset, Politicians and Athletes in Captions (PAC), which includes person names in captions in addition to relevant scene-text found on signs, labels, or other entities in the image. PAC has 1,572 images and three captions per image. A discussion on the dataset is provided in Section 4.

By training on PAC in addition to other image-caption datasets, we create a model that can naturally integrate person names into captions. The same model still performs well on previous image captioning benchmarks. Evaluation of the methods is available in Section 5.

In summary, this paper makes three primary contributions. The special tokens framework is proposed as a method to incorporate tokens from several external sources into generated text. The PAC image-captioning dataset is collected and baseline results are presented. Lastly, this paper demonstrates the first model in the literature that integrates both facial recognition and OCR into image captioning.

2 Related Work

The ubiquitous encoder-decoder architecture divides the image captioning task into two parts. The encoder acts as feature extractor and the decoder handles word generation. Early deep learning models for image captioning used CNN encoders for feature extraction from the input image as a whole (Kiros et al., 2014; Karpathy et al., 2014; Vinyals et al., 2015).

Current models rely on attention (Bahdanau et al., 2015) to generate high-quality image captions. The seminal image captioning model, Show, Attend and Tell (Xu et al., 2015), applied attention mechanism on input visual features and the previously generated word (during inference) at each time step for textual caption word generation. The majority of current state-of-the-art methods for image captioning and visual question answering benefit from the bottom-up and top-down attention mechanism (Anderson et al., 2018). Bottom-up attention, a hard attention mechanism, leverages an object detector, Faster R-CNN (Ren et al., 2015) to detect the most important regions in the image. Top-down attention, a soft attention mechanism, performs modulation over the set of input visual features from object detection regions. Following the adoption of bottom-up attention for OCR features (Hu et al., 2020), we use the same mechanism to learn to include features obtained from facial recognition. Rather than Faster R-CNN, we use RFBNet (Liu et al., 2018) for facial region detection. For facial feature extraction, we use ArcFace (Deng et al., 2019) pre-trained on MegaFace dataset (Kemelmacher-Shlizerman et al., 2016).

[2]While we focus on image captioning, our method could work for integrating non-generic terms into other vision-language tasks such as VQA or visual dialogue which we leave for future work.

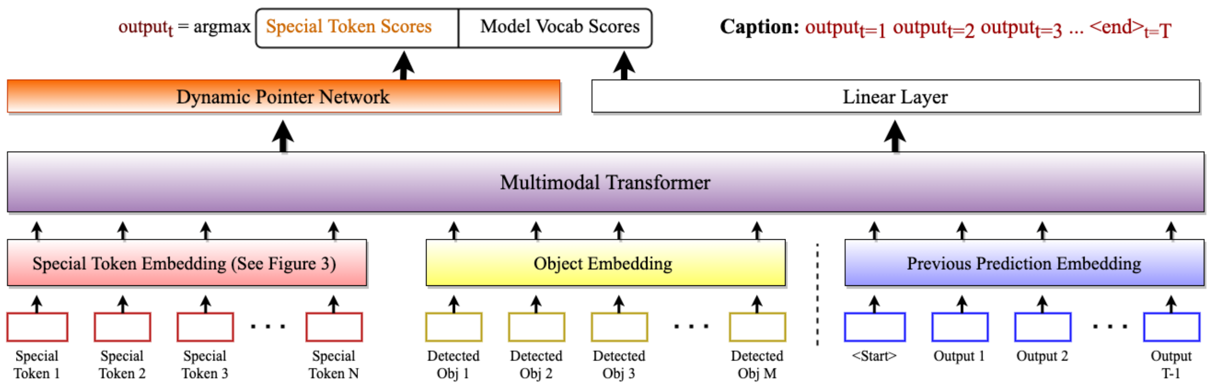


Figure 2: The architecture of the M4C + Special Tokens model. All tokens from upstream classifiers are received by the Special Token modality. The captioning model scores each vocab word and special token at each time step and outputs the highest scoring word. Our method is auto-regressive such that the caption is terminated once an *<end>* token in generated. Architecture is based on Figure 2 in Hu et al. and updated according to changes outlined in Section 3.2.

Several techniques have been proposed to handle OCR tokens in vision-language tasks. The M4C algorithm uses an indiscriminate attention layer followed by a dynamic pointer network (Hu et al., 2020). The SS-Baseline model uses individual attention blocks for each input modality followed by a single fusion encoding layer (Zhu et al., 2021). Several approaches have been proposed to better handle spatial information about OCR tokens (Gao et al., 2020b,a; Wang et al., 2021; Kant et al., 2020; Han et al., 2020; Yang et al., 2021). The MMR method utilizes spacial information about objects and scene-text via a graph structure (Mafla et al., 2021). TextOCR was introduced as an end-to-end method for identifying OCR tokens (Singh et al., 2021). TAP was introduced as a method to integrate OCR tokens into pre-training.

More similar to our work, Zhao et al. use an upstream classifier as input to a captioning model. They introduce a multi-gated decoder for handling input from external classifiers (Zhao et al., 2019). In contrast, we use general OCR and facial recognition classifiers rather than a web entity recognizer as an upstream classifier. Our approach is different from Zhao et al. in that we use bottom-up and top-down attention rather than a standalone CNN for object detection, use a common embedding space rather than a gated decoder for handling multi-modal inputs, and use rich representations (see Section 3.3) rather than only textual information for handling tokens from upstream classifiers.

MS-COCO (Lin et al., 2014) is a large dataset for common objects in context used for image captioning. Similar to MS-COCO, Flickr30k (Young et al., 2014) is another common dataset used for image captioning. Google's conceptual captions (Sharma et al., 2018) is a vast dataset used for pre-training multitasking vision-language models and fine-tuning them on other vision-language down stream tasks (Lu et al., 2019, 2020). The captions in these datasets are generic.

To facilitate use of optical character recognition in the Vision-Language domain, several datasets have been released, including ST-VQA (Biten et al., 2019) for scene text visual question answering and TextCaps (Sidorov et al., 2020) for image captioning with reading comprehension. Along with the introduction of TextCaps dataset, the M4C model (Hu et al., 2020) originally used for visual question answering was adopted for image captioning. We modify the M4C model so that it includes bottom-up facial recognition features.

3 Special Tokens

We use the term *special token* as a placeholder for extracted relevant information that is identified in an image by upstream sources. Tokens from upstream classifiers are *special* in that they often are named entities, offering unique descriptors for generic objects. For example in Figure 1, 'Bernie Sanders' is not a new object, but rather a special descriptor for an already recognized generic object (i.e. man). Likewise, 'this week' is not a generic temporal entity. Instead, it can be used to give more detail about a generic object: a screen that says 'this week', referring to a TV show or event called 'this week'.

We call our corresponding method for integrat-

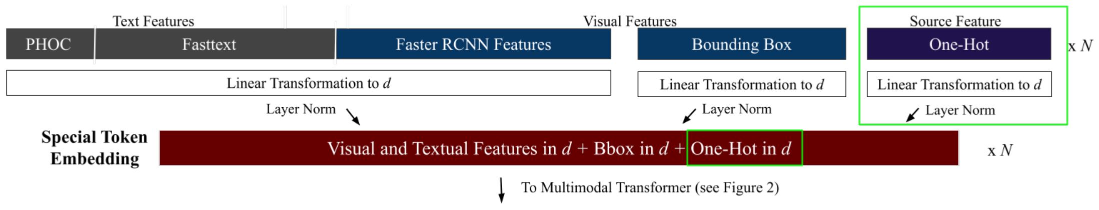

Text Features
Visual Features
Source Feature
PHOC
Fasttext
Faster RCNN Features
Bounding Box
One-Hot
x N
Linear Transformation to d
Linear Transformation to d
Linear Transformation to d
Layer Norm
Layer Norm
Layer Norm
Special Token Embedding
Visual and Textual Features in d + Bbox in d + One-Hot in d
x N
To Multimodal Transformer (see Figure 2)

Figure 3: The representation of a special token where N is the number of tokens and d is the dimensionality. We adopt the representation from Hu et al. and add the projected one-hot encoding classifier type feature (highlighted in green box). We are the first to use this representation for facial recognition tokens in addition to OCR tokens. See Equation 2 for more detail.

ing special tokens into image captions the *special token approach*. In our approach, there are two modalities that hold information about an image. The first modality corresponds to generic visual features (yellow box in Figure 2) which are responsible for informing the model of general context (all vision-language models have a visual modality). The second modality, special tokens (red box in Figure 2), is responsible for informing the model of specific terms that are relevant to the image. The embeddings for the first modality are calculated from visual features from an object detector. The embeddings from the special token modality are calculated from visual feature vectors (Faster-RCNN and a bounding box), textual features (fasttext (Bojanowski et al., 2017) and pyramidal histogram of characters (PHOC) (Almazán et al., 2014)), and a source feature (one-hot encoding) as shown in Figure 3. Additionally, special tokens are made available for direct copy into generated text which allows for zero-shot inclusion of words not seen prior. This structure has been successful on OCR vision-language datasets.

The key hypothesis of this paper is that a model can learn to differentiate tokens from separate upstream classifiers. Subsequently, the model can learn to use each token type appropriately in generated text. For example, a caption for the image in Figure 1 should neither say "A screen that says Bernie Sanders" nor should it say " 'this week' standing in front of a screen."

As mentioned in Section 1, this work demonstrates using two types of special tokens, OCR tokens and facial recognition tokens. We focus our experimentation on learning to integrate facial recognition tokens by training on the PAC dataset. However, any set of words that can be identified by some classification or recognition module can conceivably be a set of special tokens. We leave integration of more upstream vision classifiers for future work.

3.1 Trade-Offs

The goal of the special token approach is to integrate vocabulary tokens from external sources into generated text. The special tokens approach is based on several following observations.

1) Different machine learning architectures have been designed to perform well on different tasks. For example, tasks such as OCR detection and facial recognition, benefit from specialized methods that differ from traditional object detection. OCR recognizes and combines characters rather than directly classifying entire words or sentences. In facial recognition, a regression model is trained to output face embeddings which are subsequently compared to embeddings of known individuals. Even in standard classification tasks, significant research is put into fine-tuning architectures to get state-of-the-art results on dataset benchmarks. Such work can be leveraged by a captioning model by using these classifiers as upstream sources.

2) The space of all possible vocabulary tokens, when named entities or proper nouns are included, is intractably large. By appending special tokens to the vocabulary at inference time, the captioning model's vocabulary is prevented from increasing vastly.

3) Using non-generic terms does not always increase the syntactic or semantic complexity of the caption. For example in Figure 1, the name 'Bernie Sanders' is a substitution for what can also be a generic term such as 'man'. If a captioning model can generate a caption such as 'A person standing in front of a screen', the same contextual understanding should be able to generate the caption 'Bernie Sanders standing in front of a screen.' The model just needs to know to *use* the named entity 'Bernie Sanders'. The special token approach takes advantage of this by allowing the model to

learn representations for *types* of special tokens. In Section 5.3 we show that our model learns to represent different token types in different sections of the embedding space. The model can then implicitly associate sections of the embedding space with related generic objects.

4) The desired vocabulary may not be constant. For example, after an election cycle, new politicians become commonplace and a captioning model may need to adapt accordingly. The special token approach is highly practical in this sense. The captioning model does not need re-training, only the upstream facial recognition model needs to be updated.

3.2 Adopting M4C

We utilize the multimodal multi-copy mesh copy (M4C) model introduced by Hu et al. in order to copy special tokens into generated text (Hu et al., 2020). We are the first to utilize this method for tokens other than OCR. Here, we formalize the differences between our captioning model and the M4C captioning model. Figure 2 provides a corresponding architecture diagram.

The input modalities into the M4C captioning model are object features $\{x_1^{obj}, ..., x_M^{obj}\}$ for M objects and OCR tokens $\{x_1^{ocr}, ..., x_N^{ocr}\}$ for N OCR tokens. We generalize OCR tokens to special tokens st such that the inputs are $\{x_1^{obj}, ..., x_M^{obj}\}$ and $\{x_1^{st}, ..., x_N^{st}\}$ for N tokens in total. M4C captioner predicts fixed vocab scores $\{y_{1,t}^{voc}, ..., y_{K,t}^{voc}\}$ where K is a fixed vocabulary size and t is the decoding step, and OCR vocabulary scores $\{y_{1,t}^{ocr}, ..., y_{N,t}^{ocr}\}$ where N is the number of OCR tokens. The selected word at each time step $w_t = argmax(y_t^{all})$ where $y_t^{all} = \{y_t^{voc} \cup y_t^{ocr}\}$. We substitute $y_t^{st} = \{y_{1,t}^{st}, ..., y_{N,t}^{st}\}$, where N is the number of special tokens, for y_t^{ocr} such that $y_t^{all} = \{y_t^{voc} \cup y_t^{st}\}$. Special token vocabulary scores $y_{1...N,t}^{st}$ are calculated by combining linear transformations of the decoded output z_t^{dec} and the decoded special token representations z_n^{st} as shown below:

$$y_{n,t}^{st} = (W^{st} z_n^{st} + b^{st})^T (W^{dec} z_t^{dec} + b^{dec}). \quad (1)$$

3.3 Rich Representations

Several types of information may be important for determining if and how a special token should be used in generated text. This may include information about where a special token is located in an image, what the token looks like, or how the token was generated. For example, a known person in the center of an image is more likely to be relevant than a small segment of text found on a sign in the background of an image. Several features are used to richly encode these features of each special token. Hu et al. use visual, spatial, and textual features to calculate OCR tokens embeddings (Hu et al., 2020). We adopt this representation for all special tokens and add an additional source feature to differentiate the upstream classifiers used for identifying special tokens. A formal description of the special token embedding calculation is described below and a visual representation is provided in Figure 3.

Special tokens are represented by a feature vector x_i^{st}, where $i = 1...N$. x_i^{st} incorporates visual features, textual features, and a source feature. The visual features include a bounding box x_i^b and a feature vector from an object detector x_i^{fr}. Following previous work, we use a pretrained Faster-RCNN with a ResNet backbone to generate x_i^{fr} from the RoI created by the bounding box of the token. The textual features are a fasttext (Bojanowski et al., 2017) encoding x_i^{ft} and a pyramidal histogram of characters (PHOC) (Almazán et al., 2014) encoding x_i^p. The source feature x_i^s is a one-hot encoding between upstream classifiers used for generating special tokens. x_i^{fr}, x_i^{ft}, and x_i^p are concatenated together and projected onto a tuned encoding dimensionality d by a learned linear transformation W_1. Additionally, x_i^b and x_i^s are projected onto d by learned linear transformations W_2 and W_3. These transformations are trained during the same time as the captioning model. Layer normalization LN is applied to the three d dimensional vectors. x_i^{spec} is a result of element wise addition of these three vectors after layer normalization as shown below:

$$\begin{aligned} x_i^{spec} &= LN(W_1([x_i^{fr}; x_i^{ft}; x_i^p])) \\ &+ LN(W_2 x_i^b) + LN(W_3 x_i^s). \end{aligned} \quad (2)$$

3.4 Loss

We do training with decoding binary cross entropy loss $\mathcal{L}_{dbce}$ such that the model is supervised at each decoding step t with binary cross entropy $\mathcal{L}_{bce}$.

$$\mathcal{L}_{dbce} = \sum_{t=1}^{T_{end}} \frac{\mathcal{L}_{bce}(t)}{T_{end}} \quad (3)$$

where T_{end} is the number of decoding steps before $<end>$ is predicted from the vocabulary. A max-

1. Manny Pacquiao in a AIBA world championship
2. Manny Pacquiao poses for a photo at an event.
3. Boxer Manny Pacquiao is posing for a photo.

1. Megan Rapinoe moves the ball down the field.
2. Megan Rapinoe trains hard for the next World Championship.
3.Megan Rapinoe plays soccer and wears the shirt number 15.

1. Paul Kagame is getting off the plane.
2. Paul Kagame arrived.
3. Paul Kagame gets off the plan in a suit .

1. Kamala Harris is sworn in at the 2020 inauguration.
2. Kamala Harris wearing a purple coat with one hand raised.
3. Kamala Harris being sworn in and standing in front of the Capitol Building.

Figure 4: Samples from the Politicians and Athletes in Captions dataset

imum number of decoding steps T_{max} is set such that $T_{end} <= T_{max}$.

At each decoding step, sigmoid activation and binary cross entropy are applied uniformly across the fixed model vocabulary of size K and the vector of special tokens of size N such that

$$\mathcal{L}_{bce} = g_n * \log(\sigma(y_n)) + (1 - g_n)\log(1 - \sigma(y_n)) \quad (4)$$

where $n = 1...K+N$, y_n is predicted value, and g_n is expected value.

4 PAC Dataset

With this paper we create the Politicians and Athletes in Captions (PAC) dataset. PAC is image-caption dataset consisting of images of well-known individuals in context. PAC includes 1,572 images and three captions per image. Samples from PAC can be seen in Figure 4 and additional samples can be found in the supplementary materials.

We create PAC with the goal of studying the use of non-generic vocabulary in image captioning. The non-generic terms emphasized in PAC are person names and OCR tokens. The PAC dataset offers several technical challenges: 1) correctly identifying people in a variety of settings, 2) reasoning about the effect of the *presence* of the individual. If a known person is in a scene, the description of the scene is often based on the known person, and 3) natural integration of a name into a generated caption.

4.1 Collection

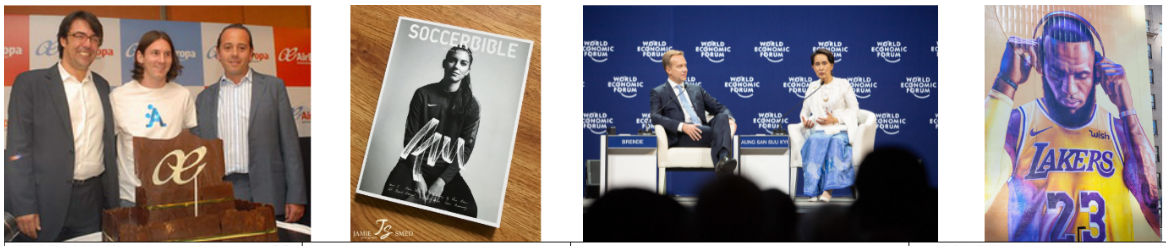
Images were collected from the Creative Commons image database which are made available under the CC licence. To find individuals for the dataset we searched for 'famous athletes' and 'famous politicians' and selected 62 individuals. The selected well-known individuals are of various races and sexes and are from many parts of the world. For image collection, we searched for each of the 62 well-known individuals and selected images by manually filtering out duplicates and images without visible faces.

Annotators were instructed to provide a caption of the image including the name of the individual which was searched for when collecting the image. Other famous individuals who happened to appear in the image may also be mentioned in the captions. Additionally, annotators were instructed to use scene-text if it improved the quality of the caption. These annotation instructions differ from those for caption collection of previous datasets. For example, in the collection of MS-COCO captions, annotators were instructed to *not* use proper nouns (Chen et al., 2015) and annotators for TextCaps were instructed to always use text in the scene (Sidorov et al., 2020). 658 images were captioned by college students and 914 were captioned by Amazon Mechanical Turk. Captions were scanned for grammar and spelling errors.

4.2 Analysis

PAC includes images 1,572 images with 3 captions each. All images include at least one famous politician or athlete. Overlap exists in several images. 62 different individuals are in the dataset for an average of 25.2 images per person. 23 of the individuals are politicians while 39 are athletes.

Each caption includes the name of at least one person name in the image. In 66.1% of images, there is scene text that is recognized by Google Cloud OCR (not all photos have scene text). For 35.9% of images, at least one of the captions uses scene text (as recognized by Google Cloud OCR). In comparison, 96.9% of TextCaps images have scene text and 81.3% of captions use scene text. In the PAC dataset, 96.3% of the images contain



M4C+ST: lionel messi poses with two other people.	**M4C+ST:** alex morgan standing on a book cover.	**M4C+ST:** aung san suu kyi is at the world economic forum.	**M4C+ST:** lebron james wearing a purple jersey with the number 23
M4C: a man and a woman are posing for a photo with the word " ropa " on it	**M4C:** jamie photography on a book cover	**M4C:** a photo of a man in front of a screen that says world world world world economic forum.	**M4C:** a close up a soccer player wearing a purple shirt with the word "lakers" on it

Figure 5: Captions generated for PAC test set images. Red words indicate tokens from the face recognition module and blue words indicate tokens from the OCR module. Corresponding metrics found in Table 1.

a face region of interest (RoI) that is detected by the RFB Net (Liu et al., 2018), the face detector we use throughout this work (Sidorov et al., 2020).

4.3 Limitations

We identify two primary limitations of the PAC dataset. The dataset with 1,572 images is small relative to similar datasets. Due to this PAC cannot represent the breadth of scenes that is found in other datasets. It is recommended to use PAC in conjunction with other dataset in order to mitigate this constraint.

The second primary limitation is narrow scene representation. The dataset is of famous athletes and politicians and therefore overrepresents scenes in which athletes and politicians are photographed. The captions also reflect this bias. For example, the word 'suit' is found in 1.82% of PAC captions while only 0.14% of TextCaps captions and 0.55% in MS-COCO. The word 'microphone' is found in 1.25% of PAC captions, 0.11% in TextCaps, and 0.05% in MS-COCO. Training on PAC combined with other datasets can mitigate this limitation while still allowing the model to learn to integrate person names as demonstrated in Section 5.

5 Experiments

In our experiments, we test the special token approach by training on PAC and TextCaps. We present baseline results on PAC. Additionally, we present a visualization for the special token embedding space.

5.1 Implementation Details

For detecting regions in the image with faces, we use RFB Net (Liu et al., 2018). For facial recognition, we use ArcFace (Deng et al., 2019). Using ArcFace, we extract facial embeddings for all individuals in the dataset. At inference, we use l_2 distance to compare new embeddings to the precalculated embeddings. For PAC, ground truth face tokens are known and used during training. The facial recognition model is not used for TextCaps images at training or inference because TextCaps annotators were not instructed to use person names in captions.

We use Google Cloud OCR for extracting OCR tokens. We set a limit at $N = 50$ for the number of special tokens. Face tokens take precedence over OCR tokens if over 50 special tokens are identified. Following previous work, we use a pretrained faster RCNN (Anderson et al., 2018) with a ResNet-101 backbone to propose RoIs and extract features for each region. A limit is set at $M = 100$ object features. For caption generation, $T_{max} = 30$ is the maximum number of decoding steps.

All experiments are performed using either PAC or TextCaps. The captions of these datasets focus on using special tokens (names in PAC, OCR in TextCaps) and are therefore suitable for testing our approach. PAC is broken up into the same 80-20 train-test split for all experiments. We use the specified training and validation sets for TextCaps (Sidorov et al., 2020).

For all training, we use a batch size of 128. We use Adam optimizer with a learning rate $1e^{-4}$ and learning rate decay of .1. We use embedding input dimensionality of $d = 768$ for inputs to the encoder.

5.2 Baseline Results

We first compare our approach (M4C+ST) against the base M4C model. Both models are pretrained on TextCaps, and then trained to convergence on PAC. By adding special tokens, we see between

Table 1: Baseline scores on the PAC dataset. Our model (M4C+ST) performs significantly better than a baseline model that does not accept special tokens. For training data, an → suggests successive training. A ratio in square brackets represents a sampling ratio for training on both datasets concurrently. We follow previous work and use five common metrics for comparing results.

			PAC Test Set Metrics				
#	Model	Training	B-4	M	R	C	S
1	M4C	TextCaps→PAC	2.1	6.4	14.3	24.6	4.3
2	M4C+ST	TextCaps→PAC	**9.1**	**14.8**	**30.4**	102.6	**18.7**
3	M4C+ST	PAC,TextCaps[1:8]	8.4	14.5	30.3	**103.7**	17.5
4	M4C+ST	TextCaps→PAC,TextCaps[1:1]	5.1	12.8	25.7	73.0	14.8

ST: Special Tokens; B-4: BLEU-4; M: METEOR; R: ROUGUE; C: CIDEr, S: SPICE

112-334% percent improvements across metrics on the PAC test set (Table 1 Lines 1,2). The vanilla M4C model only has a slight chance of using the correct name which results in poor performance on PAC.

Figure 5 shows corresponding qualitative results for the these models. We observe that our model uses person names and OCR tokens appropriately throughout the captions. The right two images demonstrate M4C+ST appropriately switching between model vocabulary, face tokens, and OCR tokens during caption generation. In comparison, the M4C model refers to people generically (i.e 'man', 'woman','player') resulting in less informative captions. In the second image, vanilla M4C incorrectly uses 'Jamie Photography' (an OCR token found in the bottom left of image) as the name of a person. More qualitative samples from these models can found in the supplementary materials.

In Table 1 Lines 3 and 4, we report scores after training on different combinations of PAC and TextCaps. We find that the training procedures from Table 1 Lines 2 and 3 are the most effective. Additionally, we find that training on PAC does not degrade performance on TextCaps. Results on the TextCaps dataset can be found in the supplementary materials.

Lastly, we test our model's ability for zero-shot use of tokens from unseen individuals. New images with people not in the PAC dataset are run through our model. Qualitatively, we observe our model is able to integrate unseen individuals into image captions. These samples can be found in the supplementary materials.

5.3 Special Token Embedding Visualization

To visualize the embeddings of special tokens, we collect all embeddings during a test set run and plot them with a t-distributed stochastic neighbor embedding (t-SNE). The t-SNE plot shown in Figure 6 allows us to visualize the 768-dimensional special token embeddings in 2-dimensions. As previously mentioned, the embeddings are calculated with Equation 2 in Section 3.3. Both face tokens and OCR tokens go through same learned linear transformation ($W_{1..3}$ from Equation 2), yet the two different token types are in distinct clusters in the embedding space. This distinction is not known to the model before training, therefore during training the model effectively optimizes $W_{1..3}$ such that embeddings from each token type are meaningfully different for the multimodal transformer. This offers explanation on our observation that our model can use each token type appropriately in generated captions.

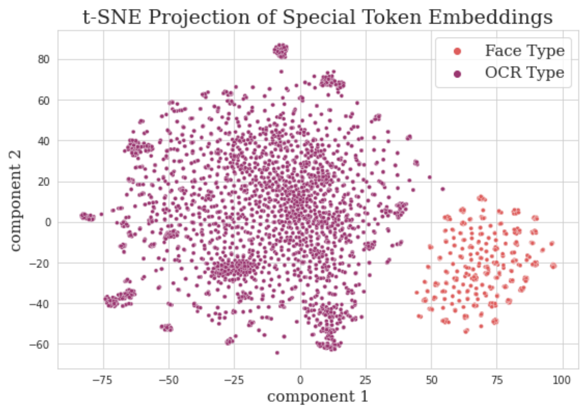


Figure 6: Projection of 768-dimensional special token embeddings into 2d space. Embeddings collected from 314 test images including 703 face tokens and 3,151 OCR tokens.

6 Conclusion

Text generated by vision-language models often lacks specific terms that would be present in human level descriptions or answers. We introduce the special token approach as an adaptable way to introduce non-generic information to a vision-language model. Our method utilizes upstream classifiers to identify information outside of generic context. The Politicians and Athletes in Captions dataset consists of image-caption pairs with well-known individuals. By using the special token approach and the PAC dataset, we train a model to integrate person names into image captions. Possible improvements to the proposed method include inclusion of more external sources or integration of open-domain knowledge with special tokens. Further progression in this direction could result in captions that are truly interesting, vivid, and useful.

Acknowledgement

The work reported in this paper is supported by the National Science Foundation under Grant No. 2050919. Any opinions, findings and conclusions or recommendations expressed in this work are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Jon Almazán, Albert Gordo, Alicia Fornés, and Ernest Valveny. 2014. Word spotting and recognition with embedded attributes. *IEEE transactions on pattern analysis and machine intelligence*, 36(12):2552–2566.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. Technical report.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*, USA.

Ali Furkan Biten, Ruben Tito, Andres Mafla, Lluis Gomez, Marcal Rusinol, Ernest Valveny, C.V. Jawahar, and Dimosthenis Karatzas. 2019. Scene text visual question answering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

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.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollar, and C Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *arXiv e-prints*, pages arXiv–1504.

Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. Uniter: Universal image-text representation learning. In *European conference on computer vision*, pages 104–120. Springer.

Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. 2019. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Chenyu Gao, Qi Zhu, Peng Wang, Hui Li, Yuliang Liu, Anton van den Hengel, and Qi Wu. 2020a. Structured Multimodal Attentions for TextVQA.

Difei Gao, Ke Li, Ruiping Wang, Shiguang Shan, and Xilin Chen. 2020b. Multi-modal graph neural network for joint reasoning on vision and scene text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12746–12756.

Wei Han, Hantao Huang, and Tao Han. 2020. Finding the evidence: Localization-aware answer prediction for text visual question answering. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3118–3131.

Ronghang Hu, Amanpreet Singh, Trevor Darrell, and Marcus Rohrbach. 2020. Iterative answer prediction with pointer-augmented multimodal transformers for textvqa. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9992–10002.

Yash Kant, Dhruv Batra, Peter Anderson, Alexander Schwing, Devi Parikh, Jiasen Lu, and Harsh Agrawal. 2020. Spatially aware multimodal transformers for textvqa. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*, pages 715–732. Springer.

Andrej Karpathy, Armand Joulin, and Li F Fei-Fei. 2014. Deep fragment embeddings for bidirectional image sentence mapping. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1889–1897. Curran Associates, Inc.

Ira Kemelmacher-Shlizerman, Steven M. Seitz, Daniel Miller, and Evan Brossard. 2016. The megaface benchmark: 1 million faces for recognition at scale. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ryan Kiros, Ruslan Salakhutdinov, and Rich Zemel. 2014. Multimodal neural language models. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 595–603, Bejing, China. PMLR.

Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, et al. 2020. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *European Conference on Computer Vision*, pages 121–137. Springer.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar, and C. Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision – ECCV 2014*, pages 740–755, Cham. Springer International Publishing.

Fen Liu, Guanghui Xu, Qi Wu, Qing Du, Wei Jia, and Mingkui Tan. 2020. Cascade reasoning network for text-based visual question answering. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 4060–4069.

Songtao Liu, Di Huang, and andYunhong Wang. 2018. Receptive field block net for accurate and fast object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems 32*, pages 13–23. Curran Associates, Inc.

Jiasen Lu, Vedanuj Goswami, Marcus Rohrbach, Devi Parikh, and Stefan Lee. 2020. 12-in-1: Multi-task vision and language representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Andres Mafla, Sounak Dey, Ali Furkan Biten, Lluis Gomez, and Dimosthenis Karatzas. 2021. Multi-modal reasoning graph for scene-text based fine-grained image classification and retrieval. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 4022–4032. IEEE.

Meredith Ringel Morris, Jazette Johnson, Cynthia L Bennett, and Edward Cutrell. 2018. Rich Representations of Visual Content for Screen Reader Users. *CHI conference on human factors in computing systems*.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. 2018. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of ACL*.

Oleksii Sidorov, Ronghang Hu, Marcus Rohrbach, and Amanpreet Singh. 2020. TextCaps: A Dataset for Image Captioning with Reading Comprehension. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12347 LNCS, pages 742–758. Springer Science and Business Media Deutschland GmbH.

Amanpreet Singh, Guan Pang, Mandy Toh, Jing Huang, Wojciech Galuba, and Tal Hassner. 2021. Textocr: Towards large-scale end-to-end reasoning for arbitrary-shaped scene text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8802–8812.

Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. 2019. Vl-bert: Pre-training of generic visual-linguistic representations. In *International Conference on Learning Representations*.

Hao Tan and Mohit Bansal. 2019. Lxmert: Learning cross-modality encoder representations from transformers. 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 5100–5111.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3156–3164, USA.

Jing Wang, Jinhui Tang, Mingkun Yang, Xiang Bai, and Jiebo Luo. 2021. Improving ocr-based image captioning by incorporating geometrical relationship. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1306–1315.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057, Lille, France. PMLR.

Zhengyuan Yang, Yijuan Lu, Jianfeng Wang, Xi Yin, Dinei Florencio, Lijuan Wang, Cha Zhang, Lei Zhang, and Jiebo Luo. 2021. Tap: Text-aware pre-training for text-vqa and text-caption. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8751–8761.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association of Computational Linguistics*, 2:67–78.

Sanqiang Zhao, Piyush Sharma, Tomer Levinboim, and Radu Soricut. 2019. Informative image captioning with external sources of information. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6485–6494.

Qi Zhu, Chenyu Gao, Peng Wang, and Qi Wu. 2021. Simple is not easy: A simple strong baseline for textvqa and textcaps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3608–3615.

Encoder-Decoder Based Image Caption Generation Framework for Assamese

Ringki Das[1] and **Thoudam Doren Singh**[1]

[1]Department of Computer Science and Engineering, NIT Silchar, India
{ringkidas,thoudam.doren}@gmail.com

Abstract

Caption generation is an artificial intelligence problem that straddles the line between computer vision and natural language processing. Although significant works have been reported in image captioning, the contribution is limited to English and few major languages with sufficient resources. But, no work on image captioning has been reported in a resource-constrained language like Assamese. With this inspiration, we propose an encoder-decoder based framework for image caption generation in the Assamese news domain. The VGG-16 pre-trained model at the encoder side and LSTM with an attention mechanism are employed at the decoder side to generate the Assamese caption. We train the proposed model on the dataset built in-house consisting of 10,000 images with a single caption for each image. We explain the experimental results in terms of quantitative and qualitative outcomes that support the usefulness of the caption generation model. The proposed model shows a BLEU score of 12.1 outperforming the baseline model.

1 Introduction

Image caption generation is a new and exciting topic in artificial intelligence that has sparked much interest and has been studied extensively in recent years. To interpret the visual contents, computer vision and natural language processing are necessary. As a result, both semantic and linguistic information about the image is required. Expressing the semantic content like human and grammatically correct is a challenging task. Caption generation has a wide range of potential real-world applications. It is helpful for visually impaired people to understand the content of the image. It can also be employed in self-driving automobiles and image search engines. This puts a halt to a slew of important real-world applications, prompting researchers to develop a better model for generating captions in the same way that humans do. Journalists can use news image captioning to describe the contents of the news as well as multimedia analytics.

Image captioning requires recognizing the important objects, attributes, and relationships in an image to generate syntactic and semantically correct description. Earlier caption generation works are based on template and information retrieval. The template-based approach generates the captions by extracting the actions, objects and other attributes in an image and filling them into a pre-defined template. In comparison, the information retrieval-based approach needs a large image database that extracts the visually similar image and generates an image caption by using the caption of the retrieved image. Nowadays, most models are based on deep learning architecture (Bai and An, 2018). The study found that the caption generated using the deep learning approach is more expressive and fluent than the traditional caption generation approaches.

Several significant image caption generation works are proposed in English using deep learning approach. According to our study, image caption generation in Assamese language is still at infancy stage. The Assamese language belongs to the Indo-European language family and it is spoken mainly in the state of Assam in India by approximately 15 million people. In this paper, we propose a caption generation model based on encoder-decoder architecture on news images. At the encoder side, a VGG-16 pre-trained model is used to represent the visual features of an image and to generate the Assamese caption, an LSTM layer with an attention mechanism is employed at the decoder side.

A large set of images and good-quality captions are required for a caption generation system. Existing English datasets are Flickr8K (Hodosh et al., 2013), Flickr30k (Plummer et al., 2015), MS-COCO (Lin et al., 2014), Lifelog (Dang-Nguyen et al., 2017), Visual Genome (Krishna et al., 2017), Multi30k (Elliott et al., 2016) etc. However, there is currently no comparable annotated dataset accessible in Assamese. Dataset scarcity is a major challenge, particularly for a morphologically rich (Saharia et al., 2012) language like Assamese. Therefore, we present an annotated dataset for caption generation in the news domain and forward it for future research. First, we collect the news articles consisting of both images and text from the local Assamese newspapers. We pre-process the data as an initial step after the collection. After that, each news image is manually annotated with one description. Furthermore, we evaluate the model using both quantitative and qualitative parameters. The proposed system is one of the earliest reported Assamese news image caption generation model and the experiment results of the developed model are promising. The objectives of this paper are:

1. The goal at hand is to build a news image caption generation model for the Assamese language in a low-resource scenario. An attention mechanism-based model is compared to the baseline model. The model is evaluated against predefined metrics to describe the news image.

2. We trained the proposed model using an in-house built dataset containing 10,000 images with single caption for each image.

2 Related Work

Fang et al. (2015) suggested a caption generation system on the MS-COCO dataset, which received a BLEU-4 score of 29.1. A visual recurrent representation model was suggested by Chen and Lawrence Zitnick (2015) for image caption generation on the MS-COCO, Flickr 8K and 30K and PASCAL 1K datasets. To describe and visualize the image caption, a bi-directional mapping between images and sentence-based descriptions was carried out. Karpathy and Fei-Fei (2015) also described the image region using a multimodal recurrent neural network on Flickr8K, Flickr30K and MS-COCO datasets. A Chinese image caption generation model was introduced by Peng and Li (2016) on Flickr30k and MS COCO dataset. They demonstrated that a character-level strategy is more effective than a word-level one. Soh (2016) reported a top-down caption generation strategy employing CNN-LSTM architecture on the MS COCO dataset with a 3.3 BLEU score. Miyazaki and Shimizu (2016) proposed a deep recurrent network based image caption generation model on the cross-lingual domain. The YJ Captions 26k Dataset, a Japanese version of the MS-COCO dataset, was built for this purpose. Amritkar and Jabade (2018) reported an image caption generation with CNN and RNN architecture on Flickr8k and MS COCO datasets.

An attention-based remote sensing image captioning system was reported by Lu et al. (2017) on their own built remote sensing caption generation dataset. They employed both hard and soft attention mechanisms to train the model. They found that the hard attention mechanism performed better than the soft attention mechanism. Dhir et al. (2019) used attention-based architecture to report a Hindi caption generation approach. They manually translated the MS COCO dataset into Hindi for the dataset. You et al. (2016) developed a semantic image attention model to concentrate on the linguistically significant image object.

Batra et al. (2018) proposed an encoder-decoder-based news image caption generating architecture on the BBC news data. The model takes an image from the news related to news documents as input and outputs an appropriate image caption.

Rahman et al. (2019) introduced Chittron, a Bangla image captioning model. A total of 16,000 images was collected and has been manually annotated by two native Bangla speakers. Next, a VGG-16 image embedding model integrated with a stack LSTM layer is used to train the model. The proposed model has gained a BLEU score of 2.5. Again Kamal et al. (2020) used deep learning techniques to create an automated Bangla caption generation system called TextMage on the BanglaLekhaImageCaptions dataset. The TextMage model could understand the visual scenes that belong to the Bangladeshi geographical context. The proposed model is trained on the BanglaLekhaImageCaptions dataset consisting of 9,154 images along with two descriptions for each image. The use of Visual Genome image captioning in a multimodal machine translation challenge was reported by Meetei et al. (2019b). The generation and evaluation of Hindi image caption on the Visual Genome dataset were carried out by Singh et al. (2021a). Additionally, attention based image and video caption generation framework were carried out by Singh et al. (2021c), Singh et al.

Table 1: Statistics of the dataset

Data set	Image	Caption
Train	8000	8000
Development	1000	1000
Testing	1000	1000
Total	10000	10000

(2021b). Meetei et al. (2019a) reported a work on identifying the Manipuri and Mizo texts in an image that is a crucial challenge in image captioning.

3 An Assamese Multimodal Dataset

Several benchmark datasets for caption generation are Flickr8K, Flickr30K and MSCOCO, available in English, but none are accessible for resource-constrained languages, including Assamese. So, we built an Assamese multimodal dataset from the news domain. We carried out the following dataset preparation steps:

1. Collection of data
2. Image pre-processing
3. Image annotation

3.1 Data collection

The preparation of a standard dataset is one of the most challenging parts of a deep neural network model. A newspaper has both images and text, making it a valuable source of information. To address the data set availability problem, we collected 10,000 Assamese news images from three Assamese local e-newspapers, namely Ganaadhikar[1], Niyomia Barta[2] and Asomia Pratidin[3]. The data is collected during June 2020 and April 2021. After pre-processing and annotation, the raw information is used to train our model. Each news image is manually annotated with an insightful narrative relevant to the news event by two native Assamese speakers. A statistics of Assamese news caption dataset is presented in Table 1.

3.2 Image pre-processing

Based on the news event, the original image is manually cropped to highlight the essential portion of the image to extract the relevant part of the image. The specific object features of images must be combined in the correct order to correlate with the caption. A sample data is shown in Figure 1 where the news is about the baby; therefore, we crop the image wherein the Figure 1B focus on the baby only.

3.3 Image annotation

Describing the image content is one of the important tasks of a caption generation model. Each image has been manually annotated with one image caption from the news content by two native Assamese speakers. At times, the content and the image cannot convey the same meaning. As a result, the annotators have labeled each image with a more appropriate news caption to the event as part of the post-editing process. Some news articles have only the logo or file images which are not relevant for the image. These irrelevant images are filtered out. Then, we put a correct news description by performing a manual post-editing of the captions for a better captioning model. Figure 2 shows one sample of the Assamese news caption dataset.

4 System Architecture

The first stage of an image caption generation model is image feature extraction, and the second part is image description generation. A convolutional neural network is used to extract the image features at the encoder side and LSTM layers are used to train the language model for image description at the decoder side. This paper describes the development of an encoder-decoder based image caption generation framework using CNN-LSTM architecture with attention mechanism in the Assamese language news domain. The proposed model consists of three phases:

1. Text pre-processing
2. Image feature extraction
3. Caption generation

4.1 Text pre-processing

Before feeding the text input to the neural network, it is important to pre-process the text data and transform it into a numerical form. For input text representation, a word embedding layer is used. It provides a dense representation of the input text and then passes it to the next LSTM layer.

[1]http://ganaadhikar.com
[2]https://niyomiyabarta.org/home/
[3]https://www.asomiyapratidin.in/

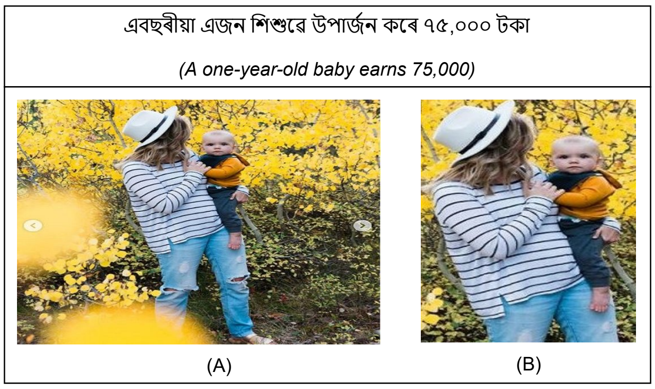

এবছৰীয়া এজন শিশুৱে উপাৰ্জন কৰে ৭৫,০০০ টকা
(A one-year-old baby earns 75,000)
(A)
(B)

Figure 1: Image cropping

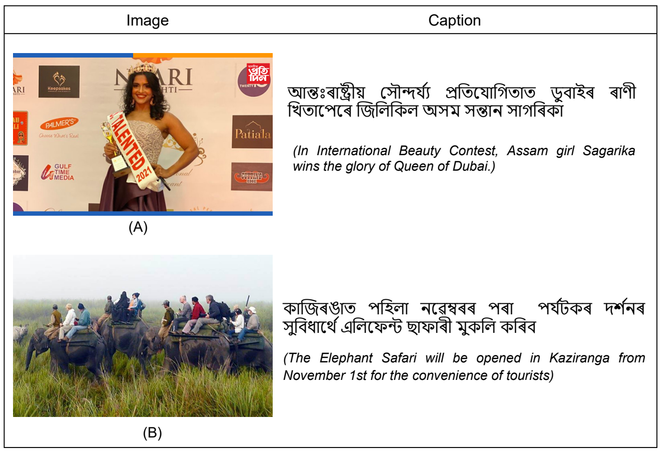


Image	Caption
(A)	আন্তঃৰাষ্ট্ৰীয় সৌন্দৰ্য্য প্ৰতিযোগিতাত ডুবাইৰ ৰাণী খিতাপেৰে জিলিকিল অসম সন্তান সাগৰিকা *(In International Beauty Contest, Assam girl Sagarika wins the glory of Queen of Dubai.)*
(B)	কাজিৰঙাত পহিলা নৱেম্বৰৰ পৰা পৰ্যটকৰ দৰ্শনৰ সুবিধাৰ্থে এলিফেন্ট ছাফাৰী মুকলি কৰিব *(The Elephant Safari will be opened in Kaziranga from November 1st for the convenience of tourists)*

Figure 2: An example of Assamese caption dataset

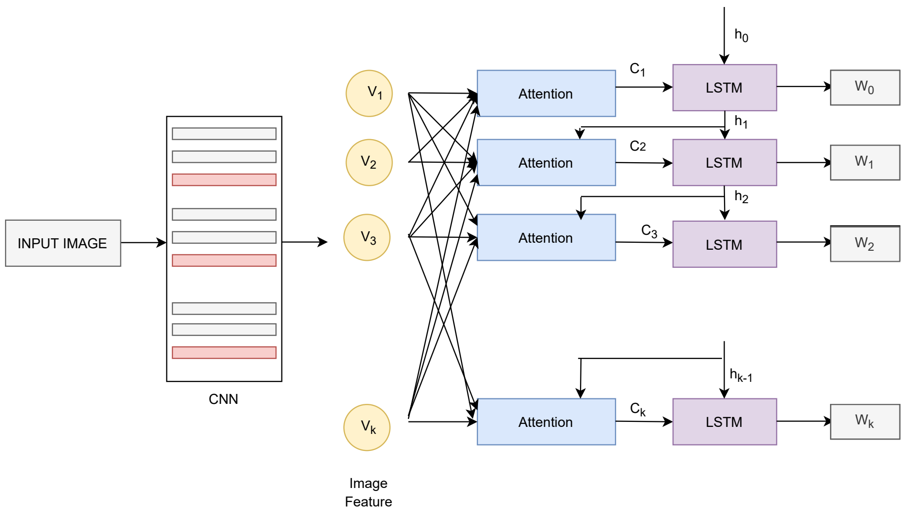


Figure 3: A graphical representation of proposed architecture

4.2 Image feature extraction

A convolutional neural network (CNN) is deployed to extract the image features. It encodes an image into an intermediate vector representation. To extract a feature set vectors, CNN is employed as an encoder. In this framework, we use VGG-16 (Simonyan and Zisserman, 2014) pre-trained CNN model as an encoder. VGG-16 model is trained on the ImageNet [4] dataset. It encodes the input image into a fixed-length vector for further processing to generate the image description. For image

[4]https://image-net.org/

feature extraction, the input image is resized into 224×224 dimensions. We discard the output of the last layer and stored the output of 6_{th} layer. The dimension of each feature vector is $7\times\ 7\ \times\ 512$.

4.3 Caption generation

To solve the vanishing gradient problem, the long short-term memory (Hochreiter and Schmidhuber, 1997) is employed as a decoder that can learn long-term dependencies. It is used for language modeling trained on text data to predict the next word. LSTM is trained so that it can produce the caption by generating one word for every time step conditioned on a context vector, the previous hidden state and the previously generated words. First, the captions are tokenized to create a lexicon of unique words. The size of our vocabulary is 8558. The model understands the start and end of each caption since each sentence is concatenated with the "start" and "end" tags.

4.4 Attention Mechanism

The fixed-length context vector in a sequence to sequence (seq2seq) model fails to remember long sentences. As a result, an attention mechanism (Xu et al., 2015) is utilized to solve this problem. The attention mechanism works on the relevant parts of the input image and ignores the rest. Rather than compressing an entire image into a static representation, attention allows the salient features to dynamically come to the forefront as needed. In simple terms, the context vector is a dynamic representation of the relevant part of the image input at time t. The attention mechanism considers the relevant part of the image when the LSTM generates a new word, so the decoder only uses that part of the image. An attention mechanism is classified into local and global attention mechanisms. Global attention is defined as paying attention to all source parts of an image (Luong et al., 2015). Local attention focuses to only a few source positions (Bahdanau et al., 2014).

In this current work, we employ a global attention mechanism that is placed in all source positions. In between CNN and LSTM, we use the attention mechanism to help the decoder to focus the important parts of the image. The global attention mechanism considers all the hidden states of the encoder while deriving the context vector c_t. In order to compute the context vector c_t, we first compute the variable-length alignment vector a_t. The variable-length alignment vector a_t whose size equals the number of time steps on the source side is derived by comparing the current target hidden state. The encoder hidden states and their respective alignment scores are multiplied to calcuate the context vector. The formula for calculating the context vector, alignment vector and score are listed in equations 1, 2, 3 and 4, respectively.

$$c_t = \bar{h}_s * a_t(s) \tag{1}$$

In equation 1, global context vector c_t is computed as the weighted average of the encoder hidden states $\bar{h}_s$ and alignment vector a_t.

$$a_t(s) = align(h_t, \bar{h}_s) \tag{2}$$

$$= \frac{exp(score(h_t, \bar{h}_s))}{\sum_s exp(score(h_t, \bar{h}_s))} \tag{3}$$

From the equations 2 and 3, the variable-length alignment vector a_t is derived by computing the similarity between current target hidden state h_t with each source hidden state $\bar{h}_s$.

$$score(h_t, \bar{h}_s) = \begin{cases} h_t^T \bar{h}_s & dot \\ h_t^T w_a \bar{h}_s & concatenate \\ v_a^T \tanh(w_a(h_t, \bar{h}_s)) & general \end{cases} \tag{4}$$

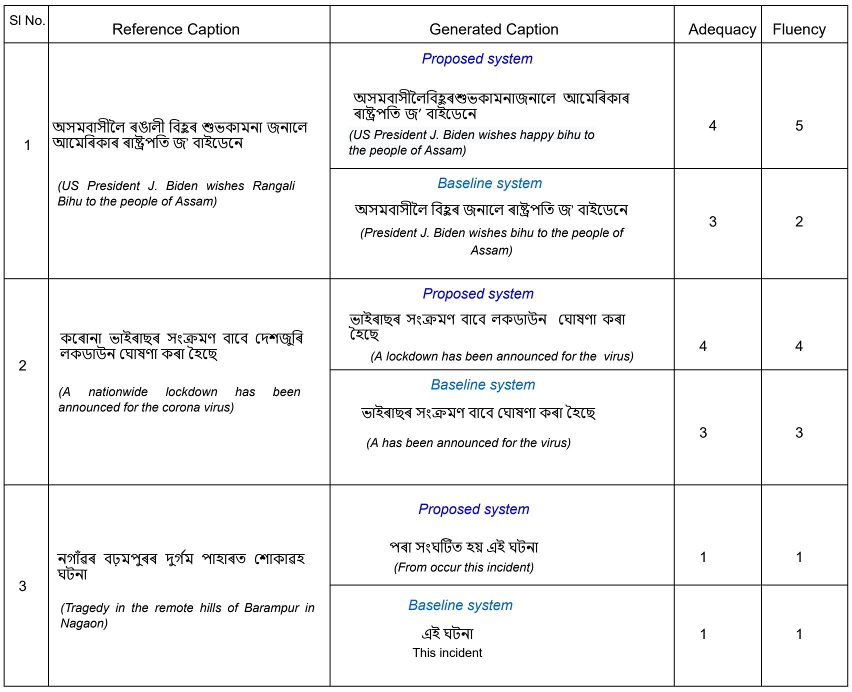
Again in equation 4, score is referred as a content-based function for which we consider three different alternatives like dot, concatenate and general, respectively.

4.5 Beam Search

Beam search is an optimization search strategy for reducing memory requirements. The search tree is built using a breadth-first search approach. We use the beam search method to evaluate the captions. For generating sentences of size t + 1, the beam search approach inspects the top k sentences and holds the highest probability one until it reaches the "end" tag or the maximum length of the caption.

Table 2: Adequacy and fluency rating scale

Rating	Adequacy	Fluency
5	All	Flawless
4	Most	Good
3	Much	Non-native
2	Little	Disfluent
1	None	Incomprehensive



Sl No.	Reference Caption	Generated Caption	Adequacy	Fluency
1	অসমবাসীলৈ ৰঙালী বিহুৰ শুভকামনা জনালে আমেৰিকাৰ ৰাষ্ট্ৰপতি জ’ বাইডেনে *(US President J. Biden wishes Rangali Bihu to the people of Assam)*	*Proposed system* অসমবাসীলৈবিহুৰশুভকামনাজনালে আমেৰিকাৰ ৰাষ্ট্ৰপতি জ’ বাইডেনে *(US President J. Biden wishes happy bihu to the people of Assam)*	4	5
		Baseline system অসমবাসীলৈ বিহুৰ জনালে ৰাষ্ট্ৰপতি জ’ বাইডেনে *(President J. Biden wishes bihu to the people of Assam)*	3	2
2	কৰোনা ভাইৰাছৰ সংক্ৰমণ বাবে দেশজুৰি লকডাউন ঘোষণা কৰা হৈছে *(A nationwide lockdown has been announced for the corona virus)*	*Proposed system* ভাইৰাছৰ সংক্ৰমণ বাবে লকডাউন ঘোষণা কৰা হৈছে *(A lockdown has been announced for the virus)*	4	4
		Baseline system ভাইৰাছৰ সংক্ৰমণ বাবে ঘোষণা কৰা হৈছে *(A has been announced for the virus)*	3	3
3	নগাঁৱৰ বড়মপুৰৰ দুৰ্গম পাহাৰত শোকাৱহ ঘটনা *(Tragedy in the remote hills of Barampur in Nagaon)*	*Proposed system* পৰা সংঘটিত হয় এই ঘটনা *(From occur this incident)*	1	1
		Baseline system এই ঘটনা This incident	1	1

Figure 4: Rating score based on adequacy and fluency

4.6 Training Details

The input of the VGG-16 convolutional neural network is 224x224 RGB images, which produces a vector of size 49x512 for each image. Again the captions are fed into the word embedding layers with 256 neurons. We train our model with 0.5 dropout rate, softmax cross-entropy loss function and Adam optimizer (Kingma and Ba, 2014) with batch size of 64 for 25 epochs. For training, we use 8000 images, and for development and testing, we use 1000 images each. The experimental results demonstrate that the LSTM with attention mechanism as a middle layer showed more effectiveness in generating the image captions.

5 Experimental Results and Discussion

The quantitative and qualitative analysis of the proposed system is covered in this section.

5.1 Quantitative analysis

For quantitative analysis, the BLEU (Papineni et al., 2002) metric is used. It checks the similarity of a generated output sentence corresponding to a reference sentence. We report the BLEU scores of the baseline and proposed models in Table 3. The formula for calculating the BLEU score is listed in equations 5 and 6.

$$BP = \begin{cases} 1 & \text{if } c>r \\ 0 & \text{if } c<=r \end{cases} \tag{5}$$

$$BLEU = BP * exp(\sum_{n}^{N} w_n \log P_n) \tag{6}$$

where,
c is candidate sentence length
r is reference sentence length
P_n is n-gram precision
w_n is weight

5.2 Qualitative Analysis

To verify the correctness of the machine-generated output, two native speakers of Assamese evaluate the generated captions by using the criterion set by linguistic data consortium(LDC) (Denkowski and Lavie, 2010). A sample example based on adequacy and fluency rating scale is shown in Figure 4 and it is found that most of the captions are flawless. According to LDC, human judgment is classified into adequacy and fluency categories (Table 2).In comparison to the source text, adequacy refers to how much meaning the target text can express. Fluency is the capability to describe a grammatically correct target text. To calculate the adequacy and fluency score, we randomly pick 100 sentences

Table 3: BLEU score of our proposed and baseline architectures

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Baseline	31.8	25.3	20.5	11.4
Proposed	40.4	31.6	21.8	12.1

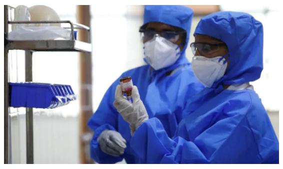
News image	Reference caption	Generated caption
(A)	নৰেন্দ্র মোদী আজি দেশৰ উদ্দেশ্যে ভাষণ দিছে *(Narendra Modi is addressing the nation today)*	*Proposed system* নৰেন্দ্র মোদীৰ সংবাদমেল *(Conference of Narendra Modi)* *Baseline system* মোদীৰ বৈঠক *(Conference of Modi)*
(B)	অগ্নিগৰ্ভা নাগালেণ্ডত আৰক্ষী নিহতৰ সংখ্যা ১৪জনলৈ বৃদ্ধি *(Firefight death toll rises to 14 in Nagaland)*	*Proposed system* ভয়াবহ অগ্নিকাণ্ড আৰক্ষী নিহত *(Horrific fire kills police)* *Baseline system* ভয়াবহ অগ্নিকাণ্ড *(Horrific fire)*
(C)	টীম ইণ্ডিয়াৰ তাৰকা বিৰাট কোহলি কঠোৰ অনুশীলনত মগ্ন হৈ আছে। *(Team India's star Virat Kohli is immersed in rigorous training.)*	*Proposed system* অনুশীলনত মগ্ন কোহলি *(Kohli immersed in practice)* *Baseline system* বিৰাট কোহলি *(Virat Kohli)*
 (D)	ভাৰতত কৰোনাত আক্রান্ত হৈ মৃতুৎ হোৱা লোকৰ সংখা বৃদ্ধি *(In India, the number of death troll due to corona is increasing)*	*Proposed system* ভাৰতত কৰোনাত মৃতুৎ লোক বৃদ্ধি *(Death toll due to corona rises in India)* *Baseline system* কৰোনা *(corona)*

Figure 5: Generated captions by the proposed and baseline models

Table 4: Human evaluation results

Model	Adequacy	Fluency
Baseline	1.48	1.96
Proposed	1.91	2.05

from the test dataset. Table 4 shows the scores for adequacy and fluency respectively.

5.3 System Comparison

Till today, no model has been reported in Assamese news image captioning to the best of our knowledge. To make a fair comparison, we propose a baseline model of CNN-LSTM architecture (Vinyals et al., 2015) and compared with the proposed model as shown by Table 3.

5.4 Discussion

Figure 5 shows the sample input and output for the image caption generating model. From the Figure 5A, the model can detect the named entity, i.e., "Narendra Modi" and also generate the caption about conference, which is a good result. The caption that is generated is meaningful, although it is less fluent. Therefore, the adequacy and fluency are considered as 4 and 3, respectively, from the point scale rating (Table 2). As shown in Figure 5B, the model can also show a good result. The model can detect the "fire" and the "army" as part of the image. The machine-generated caption can convey the meaning. In this example, the adequacy and fluency scores are 5 and 4, respectively. As shown in Figure 5C, the model identifies the named entity, i.e., "Virat Kohli" and also generated caption says about the action. Thus, the generated caption can convey the meaning and is fluent. As a result, both adequacy and fluency receive a 5 on the point scale. After seeing the "PPE kit", costume, the model can generate about the "corona" in Figure 5D. But the generated caption is not fluent. So the adequacy and fluency rating is 3 and 2, respectively.

5.5 Error analysis

There are couple of reason why the generated captions are imperfect. Poor caption quality can be a major reason for erroneous caption generation. Some image captions and news images are the least connected, which is unusual. As a result, some articles contain merely logo images or image files unrelated to current events. The absence of specific functional tokens in the training caption is another reason for poor quality generated caption.

6 Conclusion and Future Work

In this paper, we report a CNN-LSTM based framework with an attention mechanism for Assamese caption generation on the multimodal news dataset. The attention mechanism decides where to pay attention in order to generate a meaningful caption. We also report the findings of the investigation of caption generation on the Assamese language on low resource setting. To assess model performance, we used both qualitative and quantitative approaches. It is observed that the proposed framework outperforms the baseline model. In future, various architectures such as ResNet with mBERT or Indic BERT may be explored for any significant improvement of the system results further. We intend to expand the dataset in the future with a more diverse and wide collection of images of various domain-specific events, each with several appropriate descriptions, in order to build a human-like caption.

Acknowledgment

We acknowledge CNLP (Centre for Natural Language Processing) at NIT Silchar for giving access to the lab.

References

Chetan Amritkar and Vaishali Jabade. 2018. Image caption generation using deep learning technique. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pages 1–4. IEEE.

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

Shuang Bai and Shan An. 2018. A survey on automatic image caption generation. *Neurocomputing*, 311:291–304.

Vishwash Batra, Yulan He, and George Vogiatzis. 2018. Neural caption generation for news images. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Xinlei Chen and C Lawrence Zitnick. 2015. Mind's eye: A recurrent visual representation for image caption generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2422–2431.

Duc-Tien Dang-Nguyen, Liting Zhou, Rashmi Gupta, Michael Riegler, and Cathal Gurrin. 2017. Building a disclosed lifelog dataset: Challenges, principles and processes. In *Proceedings of the 15th International Workshop on Content-Based Multimedia Indexing*, CBMI '17, New York, NY, USA. Association for Computing Machinery.

Michael Denkowski and Alon Lavie. 2010. Choosing the right evaluation for machine translation: an examination of annotator and automatic metric performance on human judgment tasks.

Rijul Dhir, Santosh Kumar Mishra, Sriparna Saha, and Pushpak Bhattacharyya. 2019. A deep attention based framework for image caption generation in hindi language. *Computación y Sistemas*, 23(3).

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. 2016. Multi30k: Multilingual english-german image descriptions. *arXiv preprint arXiv:1605.00459*.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. 2015. From captions to visual concepts and back. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1473–1482.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, 47:853–899.

Abrar Hasin Kamal, Md Asifuzzaman Jishan, and Nafees Mansoor. 2020. Textmage: The automated bangla caption generator based on deep learning.

In *2020 International Conference on Decision Aid Sciences and Application (DASA)*, pages 822–826. IEEE.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Xiaoqiang Lu, Binqiang Wang, Xiangtao Zheng, and Xuelong Li. 2017. Exploring models and data for remote sensing image caption generation. *IEEE Transactions on Geoscience and Remote Sensing*, 56(4):2183–2195.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

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.

Takashi Miyazaki and Nobuyuki Shimizu. 2016. Cross-lingual image caption generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1780–1790.

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.

Hao Peng and Nianhen Li. 2016. Generating chinese captions for flickr30k images.

Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Proceedings of the IEEE international conference on computer vision*, pages 2641–2649.

Matiur Rahman, Nabeel Mohammed, Nafees Mansoor, and Sifat Momen. 2019. Chittron: An automatic bangla image captioning system. *Procedia Computer Science*, 154:636–642.

Navanath Saharia, Utpal Sharma, and Jugal Kalita. 2012. Analysis and evaluation of stemming algorithms: a case study with assamese. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, pages 842–846.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Alok Singh, Loitongbam Sanayai Meetei, Thoudam Doren Singh, and Sivaji Bandyopadhyay. 2021a. Generation and evaluation of hindi image captions of visual genome. In *Proceedings of the International Conference on Computing and Communication Systems: I3CS 2020, NEHU, Shillong, India*, pages 65–73. Springer.

Alok Singh, Thoudam Doren Singh, and Sivaji Bandyopadhyay. 2021b. Attention based video captioning framework for hindi. *Multimedia Systems*, pages 1–13.

Alok Singh, Thoudam Doren Singh, and Sivaji Bandyopadhyay. 2021c. An encoder-decoder based framework for hindi image caption generation. *Multimedia Tools and Applications*, pages 1–20.

Moses Soh. 2016. Learning cnn-lstm architectures for image caption generation. *Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, Tech. Rep*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR.

Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4651–4659.

An Efficient Keyframes Selection Based Framework for Video Captioning

Alok Singh[1], **Loitongbam Sanayai Meetei**[1], **Salam Michael Singh**[1],
Thoudam Doren Singh[1], and **Sivaji Bandyopadhyay**[1]

[1]Centre for Natural Language Processing (CNLP) & Dept. of CSE, NIT Silchar, India
{alok.rawat478,loisanayai,salammichaelcse,thoudam.doren,sivaji.cse.ju}@gmail.com

Abstract

Describing a video is a challenging yet attractive task since it falls into the intersection of computer vision and natural language generation. The attention-based models have reported the best performance. However, all these models follow similar procedures, such as segmenting videos into chunks of frames or sampling frames at equal intervals for visual encoding. The process of segmenting video into chunks or sampling frames at equal intervals causes encoding of redundant visual information and requires additional computational cost since a video consists of a sequence of similar frames and suffers from inescapable noise such as uneven illumination, occlusion and motion effects. In this paper, a boundary-based keyframes selection approach for video description is proposed that allow the system to select a compact subset of keyframes to encode the visual information and generate a description for a video without much degradation. The proposed approach uses $3 \sim 4$ frames per video and yields competitive performance over two benchmark datasets MSVD and MSR-VTT (in both English and Hindi).

1 Introduction

In recent years, we witnessed the exponential growth in multimedia data (especially video) over the Internet (Singh et al., 2019). This large volume of data creates a need for automatic video understanding systems that can describe the video's content, event and action with a short textual description. There are many applications of automatic video description generation such as efficient content indexing and searching, storytelling, the amalgamation of speech with the video description can also help visually impaired people and if the video description approaches are successful in generating a short textual description of the real-world scenes, then the robots can converse with humans effectively (Singh et al., 2020a; Aafaq et al., 2019). The task of generating image and video descriptions are very closely related. But the presence of both temporal and spatial information, which varies with the time in a video, makes the task of video description generation more challenging than image description. So for generating an informative and visually related video description, the efficient encoding of both spatial and temporal features of the video is the basic step in any video description framework.

Being an interdisciplinary problem of both computer vision and natural language processing, researchers form both domain have proposed a numerous approach for describing a video precisely, but still, much work is needed to be done. A video consists of a sequence of similar frames, but various editing effects are included in the video due to the recent advancements in technologies and these editing effects affect the process of selecting informative frames from the video. Existing approaches such as (Singh et al., 2021b; Nabati and Behrad, 2020b; Venugopalan et al., 2014; Gao et al., 2020) encode the visual features of the video either by segmenting the video in the interval of some arbitrary value k (most of time $k = 16$) or by selecting first n frames. Meanwhile, the process of encoding visual features by equal interval sampling does not guarantee that all the selected frames are informative because, in a video it is possible that the selected frames are suffering from different types of noise such as uneven illumination, motion blur, occlusion and object zoom-in/out effects (Chen et al., 2018). In this paper, we address the issue of selecting informative frames by using color information based shot boundary detection followed by keyframe selection from each shot. A shot in a video is a set of continuous similar frames captured uninterruptedly and when the content of these frames get changed, it creates two types of boundaries (transitions) in the video namely - abrupt and gradual transition. The novel contribution of the proposed work are:

i. We propose a plug-and-play keyframe selection module based on visual color information of the input frames by employing a long video temporal segmentation algorithm. This module is designed by considering the three basic requirements of any video understanding model: flexibility, efficiency and effectiveness.

ii. In the proposed framework, a temporal soft attention mechanism is employed that will focus more on the responsible keyframes from the set of selected keyframes for an input video at every time step.

iii. We perform a detailed qualitative and quantitative analysis on the results generated by the framework for MSVD, English MSR-VTT[1] and Hindi MSR-VTT[2] datasets.

The organization of the remaining part of the paper is as follows. Section 2 report a review of related work on video description. Section 3 discussed the proposed approach. A detail experimental studies is reported in Section 4 followed by conclusion in Section 5.

2 Related Work

Earlier, the process of bridging the gap between visual content and natural language was considered a challenging task. However, with the success of deep learning approaches in recent years, the gap has been reduced. Till now, the approaches proposed for video description can be categorised into three phases: classical method based phase, statistical method based phase and deep learning-based phase (Aafaq et al., 2019). Further, the related work in this section is divided into three subsections based on the type of approach is employed: Sequence-to-Sequence based approaches (S2S), attention-based approaches and boundary-based approaches.

[1]http://ms-multimedia-challenge.com/2017/challenge.

[2]https://github.com/alokssingh/MSR-VTT-Hndi-captionig

2.1 S2S video description approaches

In the early stage of the video description task, most of the approaches proposed for video description are motivated by image description approaches (Singh et al., 2021b). The pioneering work in video description is based on the prediction of SVO (Subject, Verb and Object) and fill them into a predefined templates (Aafaq et al., 2019; Singh et al., 2020a). Recently, the encoder-decoder based framework gains more popularity. Venugopalan et al. (2014) proposed a sequential Convolutional Neural Network (CNN) and Long Short Term Memory based model (CNN-LSTM) for video description. In this framework, Venugopalan et al. (2014) extracted frame-level features for each sampled frame (1 in every ten frames) using a pre-trained model and then passed all the extracted features through a mean pooling layer to get a single vector representation for the whole video. Finally, a description for an input video is generated by employing a two stacked LSTM. Although the proposed approach outperforms the previous SVO based baseline models, the model has few drawbacks such as, it does not preserve the temporal relationship among the frames and represent the whole video with a single features vector which reduces the task of video description to image description due to which lots of vital visual information get lost. To address the issues of previous model Venugopalan et al. (2014) proposed a end-to-end sequential model (Venugopalan et al., 2015) which consists of two LSTM layer. The first LSTM layer encodes the extracted visual features and the second LSTM layer receives the null padded input word concatenated with hidden representation from the first layer and generates an output description. Using a multi-stage refining algorithm (Nabati and Behrad, 2020a) proposed video description framework with content-oriented beam search. This approach involves three stages, namely feature extraction, content-oriented beam search and sentence refining. Wang et al. (2020) proposed a sequential model for encoding spatio-temporal visual representation. Unlike other sequential frameworks in this model, the sequential frame is encoded at every time step and generates the most related word at each step. In this approach, a *"Real-Time Encoder"* is introduced that uses history information of previous time steps to extract informative spatio-temporal visual representation. Recently, the work on describing a visual entities into multiple languages gained more popularity with the Hindi image captioning (Singh et al., 2021c,a), multi-modal machine translation (Meetei et al., 2019; Singh et al., 2021d) and the release of novel Video to Text (VATEX) (Wang et al., 2019) multilingual dataset (including Chinese and English) for video description. Furthermore, Singh et al. (2020b) proposed a pLSTM framework in the VATEX video captioning challenge. In this framework, two parallel LSTM are employed, which receives the input in different manners. The pLSTM framework was unable to outperform the baseline VATEX model (Wang et al., 2019) in the VATEX dataset.

2.2 Attention based approaches

On observing the effectiveness of soft attention (Xu et al., 2015) and bottom-up, top-down attention (Anderson et al., 2018) in generating visually related words at every time step in image captioning, some approaches based on attention are also proposed in the video description. Yao et al. (2015) proposed an approach that utilizes both temporal and spatial structure of the video for extracting visual features. They employed a temporal attention mechanism for selecting a relevant segment from the video. This approach only considers the first 240 frames of the video, which is the shortcoming of the proposed approach. A hierarchical Recurrent Neural Network h-RNN is proposed by Yu et al. (2016), it exploited the temporal and spatial attention for extracting visual features using Gated Recurrent Unit (GRU). Few other attention-based video captioning frameworks are proposed in (Li et al., 2018; Xiao et al., 2020). Apart from temporal attention, semantic attention is also used for generating temporally and semantically correct video descriptions. Gao et al. (2020) and Xu et al. (2019) proposed a method for video description by exploiting the combination of both semantic and temporal attention. Recently, Singh et al. (2021b) proposed hybrid attention mechanize for Hindi video captioning by utilizing the concept of visual sentinel gate (Lu et al., 2017) proposed for image captioning. The approach proposed in Singh et al. (2021b) differs from Lu et al. (2017) in terms of the implementation of the attention block.

2.3 Boundary aware approaches

An open domain video contains many editing effects, which generates a large number of shots in a video. A video consists of a large number of redundant frames and to minimise the redundancy and improve the computation time, various boundary aware approaches are proposed. (Baraldi et al., 2017) proposed a novel LSTM cell for detecting the temporal boundaries in a video and generates a visual feature vector for the whole video. (Shin et al., 2016) proposed SBD based method for the generation of the multiple sentence video description. In this method, the video is divided into shots by employing sliding windows of different lengths. Based on the assumption that selection of informative frames can improve generated description and reduce computational time (Chen et al., 2018) proposed a plug-and-play PickNet model for selecting relevant frames using reinforcement learning, then finally descriptions are generated for each video. (Sah et al., 2020) proposed a video description approach for a video surveillance system. In this approach for the multi-stream hierarchical video description model, a recurrent layer with a soft attention mechanism is employed with dynamical detected abrupt transitions. Real-time analysis is performed in support of the statement that a video description model could be useful for a video surveillance system. Few other recently proposed boundary-based video description approaches are (Shi et al., 2020; Jin et al., 2020).

3 Proposed Approach

The proposed approach consists of two modules: Boundary detection and Keyframe selection module (Sec 3.1) and Description generation module (Sec 3.2).

3.1 Boundary detection and keyframe selection phase

The main objective of boundary detection is to spot the position at which the content of the video gets changed. In this paper, we are focusing on spotting these abrupt transitions. A color histogram-based approach proposed in Mas and Fernandez (2003) is adopted to detect the temporal discontinuity in a video. The color histogram-based approach is computationally efficient and prevalent in various computer vision-related tasks. In boundary detection and keyframe selection algorithm initially, the color histogram of each frame is computed and then the histogram difference (Δ_i) is computed between the histogram of consecutive frames using Equation 1 where M is the number of bins and h_i is the color histogram of i^{th} frame in a video se-

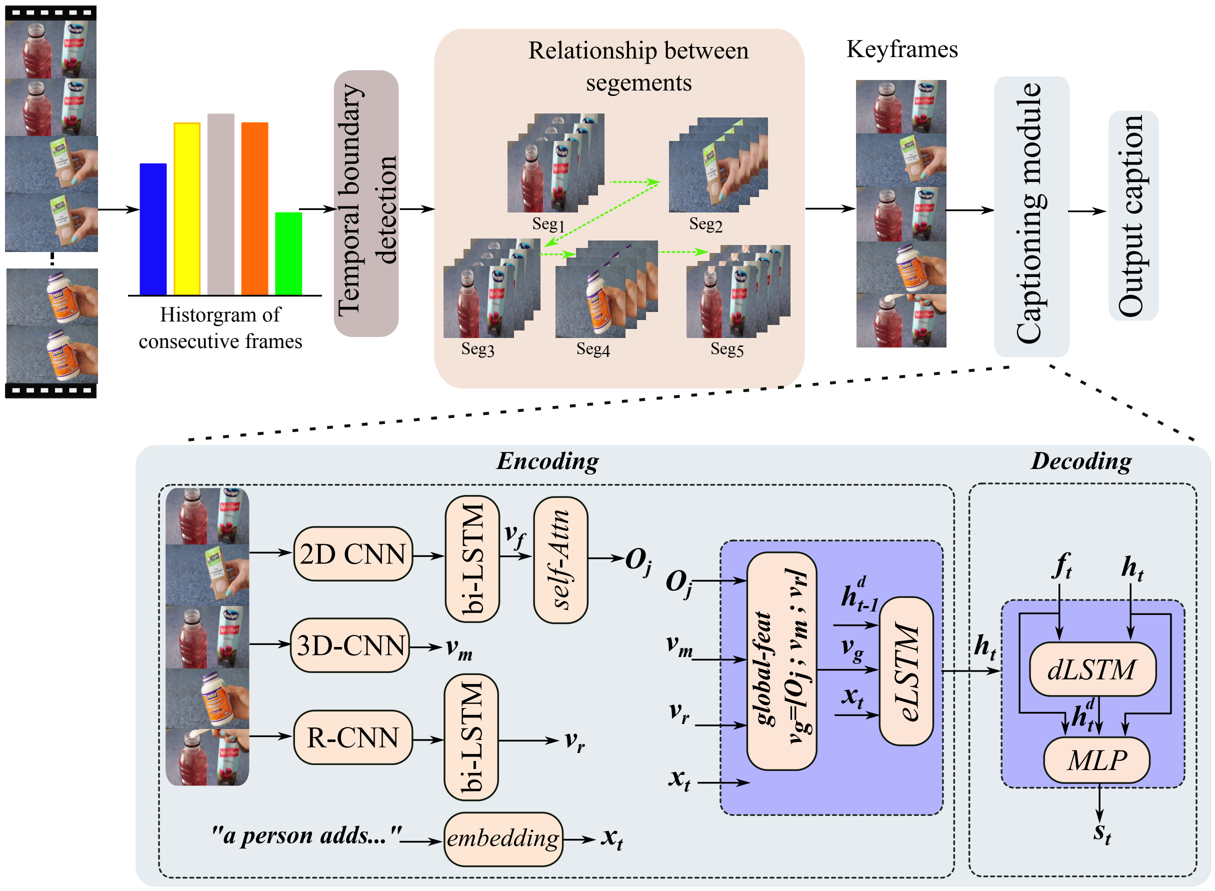


Figure 1: Pictorial representation of whole boundary based video description framework

quence.

$$\Delta_i = \left(\sum_{j=1}^{M} (h_i(j) - h_{i-1}(j))^2 \right)^{\frac{1}{2}} \tag{1}$$

After the computation of histogram difference, to declare temporal boundary at a particular location an adaptive threshold γ ($\gamma = mean(\Delta) + k \times stdev(\Delta)$) employed in Singh et al. (2019) is used, here the value of constant k is set to 5.2 after fine tuning. The mathematical expression for the declaration of temporal boundary is shown in Equation 2, where B_i record the boundary locations.

$$B_i = \begin{cases} i, & (\Delta_i \geq \gamma)\&\&(\Delta_i > \Delta_{i-1}) \\ & \&\&(\Delta_i > \Delta_{i+1}) \\ continue, & Otherwise \end{cases} \tag{2}$$

Keyframe selection: After detecting the temporal boundaries, a video is divided into different segments containing similar frames within it. A simple and computationally efficient approach for video description is the utilization of information present in keyframes of the video rather than using several redundant frames. In the proposed approach, a keyframe is selected from each segment which we get after temporal segmentation. The frame which is selected as a representative frame has a minimum distance to the other frames present in the same shot (segment). This approach is also adopted by Li et al. (2017) for video summarization. Mathematically, it can be described as follow:

$$\min_{i \in [1, n_f]} \left(\sum_{t=1, t \neq i}^{n_f} Euclidean(\tilde{h}_i - \tilde{h}_t) \right) \tag{3}$$

Where n_f is the number of frames in a shot, $\tilde{h}_i$ is color histogram of the selected frame and $\tilde{h}_t$ represent the histogram of other frames within the shot. In this way, a keyframe of each shot is selected based on the visual similarities within the shot.

3.2 Description generation phase

After selecting keyframes for an input video of N frames, we extracted three types of features that are visual appearance features (v_f) which are ex-

Algorithm 1 Temporal segmentation of video with key frame selection

```
Input: Video, V
Output: Boundaries, $Keyframes$
 1: procedure shot_detection($V$)
 2:     $F \leftarrow cv2.VideoCapture(V)$
 3:     $hist_0 \leftarrow cv2.calcHist(vid.read(F(0)), ch, m, h_s, r)$
 4:     for $i = 1$ to $length(F)$ do
 5:         $hist_i \leftarrow cv2.calcHist(vid.read(F(i)), ch, m, h_s, r)$
 6:         $\Delta_i \leftarrow \left(\sqrt{\sum_{j=1}^{M}(hist_i(j) - hist_{i-1}(j))^2}\right)$
 7:         $hist_{i-1} \leftarrow hist_i$
 8:     for $i = 1$ to $length(\Delta) - 1$ do
 9:         if $\Delta_i \geq \gamma \&\& (\Delta_i > \Delta_{i-1}) \&\& (\Delta_i > \Delta_{i+1})$ then
10:             $B_i \leftarrow record\ i^{th}$
11:          else
12:             $continue$

 1: Function $Keyframe\_sel(frames, B)$
 2: for $k = 1$ to $S$ do                                   ▷ i, j ∈ [1,nf], i≠ j
 3:     for $i = 1$ to $n_f$ do
 4:         for $j = 1$ to $n_f$ do
 5:             $diff_{(i,j)} \leftarrow record\ total\ dissimilarity\ difference$
 6:     $Keyfrm[k] \leftarrow min(diff_{(i,:)})$
 7: $return\ Keyfrm$
```

tracted using 2D CNN (He et al., 2016a), motion features (v_m) using 3D CNN and object features (v_r) extracted using R-CNN (Ren et al., 2015). Then, the appearance and object features (v_f and v_r) are post-processed using Bi-LSTMs.

3.2.1 Context rich encoding with self attention

Since a video has multiple actions and events, so some the events in earlier frames are responsible for the occurrence of other related events in forthcoming frames. Considering this fact, the post-processed appearance features are passed through a self-attention layer to get more context rich encoded visual features. Self-attention allows the model to look at the visual features of other selected keyframes for better visual encoding. So, initially using the visual features v $(v = vf)$ the value of key $K(v)$, value $V(v)$, and query $Q(v)$ are computed using Equation 4 where W_k, W_q and W_v are the weight metrics to be trained.

$$K(v) = W_k v \quad V(v) = W_v v \\ and \quad Q(v) = W_q v \tag{4}$$

Then, to compute context-rich self-attention feature maps $(O_{j \ldots S})$ dot-product attention is applied as follow:

$$\mathbf{O}_j = W_g\left(\sum_{i=1}^{S}\alpha_{i,j}V(v_i)\right) \\ where, \quad \alpha_{i,j} = softmax(\frac{K(v)^T Q(v)}{d_k}) \tag{5}$$

In the Algorithm 1, ch = channels, m = mask, h_s = histSize and r = ranges

In the above equations, S is total number of shots (segments), $v_f \in \mathbb{R}^{S \times l}$, $W_{k,v,q,g} \in \mathbb{R}^{l \times \tilde{l}}$ and the dimension of $K(v)$, $V(v)$ and $Q(v)$ is set to 64 and $d_k = 8$ following the work of Vaswani et al. (2017) for effectiveness of Self attention mechanism. For the encoding of words in the reference caption, the dense embedded representation which is obtained from a word embedding layer is passed to an encoder LSTM (eLSTM). The eLSTM takes the word embedding of input word (x) at current time step, global visual features (v_g) and decoder LSTM's hidden state of last time step as shown Equation 6.

$$h_t = eLSTM(x_{t-1}, v_g, h_{t-1}^d) \tag{6}$$

3.2.2 Decoder

After getting encoded contextually rich representation of input word (h_t) and visual appearance features (O_j) they are passed to the decoder along with motion features (v_m) and object features (v_r). Before passing the self attentive appearance features (O_j) and object features (v_r) to decoder LSTM (dLSTM) they are passed through an attention layer $(Attn(V_x, h))$ as shown in Equation 7 where V $(V = O_i \ or \ v_r)$ is encoded features and W_* $(* = h, v)$ are trainable weights and b_t is bias.

$$Attn(V_x, h) = \phi(V_j, \alpha_i) \quad where, \ \phi = \sum_{i=1}^{k}\alpha_i V_{i,j} \\ and, \quad \alpha_i = softmax(W_a tanh(W_v V + W_h h_{t-1} + b_t)) \tag{7}$$

After getting attentive appearance features and object features from the attention layer they are concatenated with motion feature (v_m) and passed to decoder LSTM (dLSTM) as shown in Equation 8 where $[;]$ denotes concatenation and h_t^d is used in Equation 6.

$$f_t = [Attn(O_j, h_t); Attn(v_r, h_t); v_m] \\ h_t^d, c_t^d = dLSTM([f_t; h_t]) \tag{8}$$

Further, the word probability s_t at every time step is decoded as follow:

$$s_t = softmax(MLP([f_t; h_t^d; h_t])) \tag{9}$$

The cost function used for maximizing the likelihood of the correct word and minimizing the loss of the model is given by Equation 10.

$$Loss = -\sum_{t=0}^{T}\log Pr(s_t|s_{t-1}, \ldots s_0; F) \tag{10}$$

Table 1: Results of proposed approach on MSVD dataset and its comparison with other approaches.

Methods	BLEU-4	METEOR	CIDEr	ROUGE
Mean pooling				
-$AlexNet$ (Venugopalan et al., 2014)	31.20	26.90	-	-
-$AlexNet$ $(COCO)$ (Venugopalan et al., 2014)	33.30	29.10	-	-
Attention				
- SA (Yao et al., 2015)	40.28	29.00	-	-
-MMN (Li et al., 2018)	48.00	31.60	68.80	-
-$BP-LSTM$ (Nabati and Behrad, 2020b)	42.90	32.00	62.20	68.30
Boundary + Attention				
-$Boundary-aware$ (Baraldi et al., 2017)	42.50	32.40	63.50	-
-$PickNet$ (Chen et al., 2018)	46.10	33.10	69.20	69.20
-MHB (Sah et al., 2020)	43.00	**33.20**	71.10	68.70
Proposed (v_f)	45.55	30.37	68.73	66.44
Proposed $(v_f$+$v_m)$	48.66	29.90	68.33	65.97
Proposed $(v_f$+ v_m+ $v_r)$	**50.75**	32.50	**71.13**	**70.44**

4 Experimental result and discussion

4.1 Datasets

To manifest the effectiveness of the proposed approach, three benchmark datasets are employed that are: *Microsoft research video description corpus (MSVD)* (Chen and Dolan, 2011), *English Microsoft research video to text (MSR-VTT)* (Xu et al., 2016) and *Hind Microsoft research video to text (hi-MSR-VTT)* (Singh et al., 2021b). The hi-MSR-VTT dataset is recently released dataset for motivating the research on generating video descriptions in the native language. The MSVD dataset include $1,970$ videos with on average 40 descriptions for each video while the en-MSR-VTT and hi-MSR-VTT dataset include $10K$ videos with corresponding 20 descriptions. Table 2 reports the detailed statistics of all the datasets.

Table 2: Detail statistics of all the datasets

Datasets	#Training videos	#Val videos	#Test videos
MSVD	1200	100	670
MSR-VTT	6513	497	2990
hi-MSR-VTT	6513	497	2990

4.2 Metrics

For the validation of the generated descriptions, we employs Bilingual Evaluation Understudy (BLEU) (Papineni et al., 2002), Metric for Evaluation of Translation with Explicit Ordering (METEOR[3]) (Banerjee and Lavie, 2005), Consensus-based Image Description Evaluation (CIDEr) (Vedantam et al., 2015) and Recall Oriented Understudy of Gisting Evaluation (ROUGE-L) (Lin, 2004). For generating the scores for above discussed automatic evaluation metrics Microsoft COCO[4] toolkit is employed.

4.3 Parameter setting and model implementation

As discussed in section 3.2 for experimentation we employ $ResNet152$ (He et al., 2016b) as 2D CNN model for extracting appearance features of keyframes and C3D model (Karpathy et al., 2014; Tran et al., 2015) as 3D CNN for extracting the motion features. For extracting the region features Faster-RCNN (Ren et al., 2015) trained by (Anderson et al., 2018) is employed, this model extract 36 region features for each keyframes. The model is trained with $ADAM$ optimizer with learning rate 1e-4 and the learning rate is divided by 10 at every 10^{th} epoch. The number of LSTM hidden units is set to 512 and during training, the model having the best $METEOR$ score is saved. To avoid overfitting, a dropout of 0.3 is employed. In the proposed work, we tried to search optimal parameters that work comparatively better than other baseline models in all the datasets, which will minimize the time and effort required to search the best parame-

[3]The METEOR score for Hindi text is generated using: https://github.com/anoopkunchukuttan/meteor_indic

[4]https://github.com/tylin/coco-caption

Table 3: Results of proposed approach on en-MSR-VTT dataset and its comparison with other approaches.

Methods	BLEU-4	METEOR	CIDEr	ROUGE
Mean/Max pooling				
-$LSTM - GAN$ (Yang et al., 2018)	36.00	26.10	-	-
Attention				
-M^3 (Wang et al., 2018)	38.13	26.58	-	-
-MMN (Li et al., 2018)	37.50	26.40	-	-
-$ReBiLSTM$ (Bin et al., 2018)	33.90	26.20	-	-
-$BP - LSTM$ (Nabati and Behrad, 2020b)	36.60	27.00	40.50	58.70
-$MCTA$ (Wei et al., 2020)	38.50	26.90	**43.70**	-
Boundary + Attention				
-$Boundary - aware$ (Baraldi et al., 2017)	36.80	26.70	41.20	58.50
-$PickNet$ (Chen et al., 2018)	**38.90**	**27.20**	42.10	**59.50**
Proposed (v_f)	35.42	25.21	35.36	57.83
Proposed (v_f+v_m)	35.95	25.39	35.66	57.38
Proposed (v_f+ v_m+ v_r)	37.18	26.17	40.90	59.41

ters according to the dataset. All the parameter settings are the same throughout the experimentation for all the datasets. A beam search approach with beam size 7 is employed during testing to generate the final description.

4.4 Results and discussion

Comparison with existing methods: To analyse the performance proposed keyframe based video captioning approach we compare proposed approach with existing methods. For the better understanding and fair comparison all the existing methods are categorised into three type of captioning approaches that are *mean/max pooling*, *attention* and *boundary+attention*. The approaches such as *AlexNet*, *LSTM-GAN* and *pLSTM* are mean/max pooling based approaches, *MMN*, *BP-LSTM*, *M*^{3}, *ReBiLSTM* and *MCTA* are attention based while the *PickNet*, *Boundary-aware* and *MHB* are boundary based approaches which employ attention as well.

Table 1, 3 and 4 report quantitative results on MSVD, en-MSR-VTT and hi-MSR-VTT datasets. Our proposed approach outperforms other existing methods on the MSVD and the hi-MSR-VTT dataset, on 3 out of 4 metrics by a reasonable margin. While on the en-MSR-VTT dataset, our model reports comparable scores, although the *PickNet* model reports high scores, but in terms of the average number of frames used to achieve competitive performance, the proposed approach outperforms *PickNet* model. Our model uses $3 \sim 4$ frames per video whereas the *PickNet* model employ $6 \sim 8$ frames per video.

Ablation study: The proposed approach consist of two stage: boundary detection and description generation phase. To evaluate the effectiveness of all the employed visual features the proposed model is experimented with different variations such as with only appearance features, with appearance and motion features and with all three appearance (v_f), motion (v_m) and region features v_r. Table 1, 3 and 4 reports the score of proposed model with all the variation. The effectiveness of proposed method increases when all three features are employed which can be clearly seen in table 1, 3 and 4. In order to validate that whether the proposed model generates more fluent and adequate description along with high automatic scores, we perform a qualitative analysis. Figure 2 shows the description generated by the proposed model along with the output generated by $BP - LSTMs$ and ground truth (GT). On observing the output generated by the proposed model for the videos shown in Figure 2, it is clear that the keyframes based approach can generate better description than $BP - LSTM$, which employ n frames for visual encoding.

4.4.1 Analysis of picked keyframes

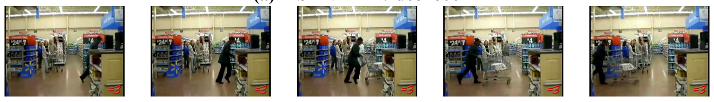
We also analysed the efficiency of the boundary based keyframe selection algorithm for selecting the most representative frame from multiple segments of the video. Figure 3 shows the distribution of keyframes selection for both the datasets. From Figure 3 it is observed that for the majority of videos, less than 8 frames are picked as a keyframe

Table 4: Results of proposed approach on hi-MSR-VTT dataset and its comparison with other approaches.

Methods	BLEU-4	METEOR	CIDEr	ROUGE
Mean/Max pooling				
-$pLSTM$ (Singh et al., 2020b)	26.10	33.00	28.50	51.20
Attention				
-$VA + SA$(Singh et al., 2021b)	36.20	39.30	**36.90**	59.80
-RNM (Tan et al., 2020)	38.80	39.10	36.00	60.70
Proposed (v_f)	34.02	38.40	30.76	58.09
Proposed (v_f+v_m)	36.11	39.95	31.12	58.95
Proposed (v_f+ v_m+ v_r)	**41.01**	**44.10**	32.85	**60.80**

BP-LSTMs: A man is doing tennis match
Our: two men are playing table tennis
Our hi: एक आदमी टेनिस खेल रहा है
GT: two men compete in a game of table tennis

(a) MSR-VTT - video7600



BP-LSTMs: a man is going through a room
Our: a man is walking
GT: a man is walking with trolley

(b) MSVD - video1929

Figure 2: Sample videos selected from each dataset with their ground truth (GT) and generated output

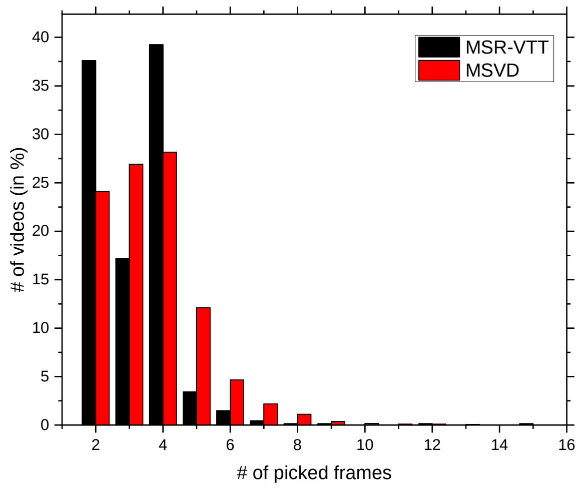


Figure 3: Statistic of picked keyframes for both the datasets

which is due to shorter video length. A video can have a single shot or multiple shots. For a single-shot video, 4 keyframes are selected at the interval of 16 and for a multi-shot video, the keyframe is selected using an approach discussed in section 3.1. From Figure 3, it is clearly observed that around 39% and 28% of videos in MSR-VTT and MSVD respectively, are single-shot videos. The average number of keyframes selected per video is $3 \sim 4$ for both MSVD and MSR-VTT dataset, which helps in avoiding unnecessary visual encoding of redundant frames and signify the efficiency of the proposed approach. Sample examples of picked keyframes are included in supplementary file.

5 Conclusion

In this paper, we employ a boundary-aware keyframe selection framework that acts as a plug-and-play module for downstream video-related tasks, such as video description and video classification. The objective of the boundary aware keyframe selection framework is to select a compact subset of keyframes for input video, which minimises the unnecessary processing of visually similar frames and ensures no degradation in the quality generated description. In the proposed approach, $3 \sim 4$ frames are selected for an input video, which is more efficient than the existing $PickNet$ model, which picks $6 \sim 8$ frames for each video. The experimental results show that the keyframes-based approach can outperform existing methods by picking keyframes and extracting different visual features such as appearance, motion and region features.

References

Nayyer Aafaq, Ajmal Mian, Wei Liu, Syed Zulqarnain Gilani, and Mubarak Shah. 2019. Video description: A survey of methods, datasets, and evaluation metrics. *ACM Computing Surveys (CSUR)*, 52(6):1–37.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086.

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.

Lorenzo Baraldi, Costantino Grana, and Rita Cucchiara. 2017. Hierarchical boundary-aware neural encoder for video captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1657–1666.

Yi Bin, Yang Yang, Fumin Shen, Ning Xie, Heng Tao Shen, and Xuelong Li. 2018. Describing video with attention-based bidirectional lstm. *IEEE transactions on cybernetics*, 49(7):2631–2641.

David Chen and William B Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pages 190–200.

Yangyu Chen, Shuhui Wang, Weigang Zhang, and Qingming Huang. 2018. Less is more: Picking informative frames for video captioning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 358–373.

Lianli Gao, Xuanhan Wang, Jingkuan Song, and Yang Liu. 2020. Fused gru with semantic-temporal attention for video captioning. *Neurocomputing*, 395:222–228.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016a. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016b. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer.

Tao Jin, Siyu Huang, Ming Chen, Yingming Li, and Zhongfei Zhang. 2020. Sbat: Video captioning with sparse boundary-aware transformer. *arXiv preprint arXiv:2007.11888*.

Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732.

Jiatong Li, Ting Yao, Qiang Ling, and Tao Mei. 2017. Detecting shot boundary with sparse coding for video summarization. *Neurocomputing*, 266:66–78.

Wei Li, Dashan Guo, and Xiangzhong Fang. 2018. Multimodal architecture for video captioning with memory networks and an attention mechanism. *Pattern Recognition Letters*, 105:23–29.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2017. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 375–383.

Jordi Mas and Gabriel Fernandez. 2003. Video shot boundary detection based on color histogram. In *TRECVID*.

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

Masoomeh Nabati and Alireza Behrad. 2020a. Multi-sentence video captioning using content-oriented beam searching and multi-stage refining algorithm. *Information Processing & Management*, 57(6):102302.

Masoomeh Nabati and Alireza Behrad. 2020b. Video captioning using boosted and parallel long short-term memory networks. *Computer Vision and Image Understanding*, 190:102840.

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.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99.

Shagan Sah, Thang Nguyen, and Ray Ptucha. 2020. Understanding temporal structure for video captioning. *Pattern Analysis and Applications*, 23(1):147–159.

Xiangxi Shi, Jianfei Cai, Jiuxiang Gu, and Shafiq Joty. 2020. Video captioning with boundary-aware hierarchical language decoding and joint video prediction. *Neurocomputing*, 417:347–356.

Andrew Shin, Katsunori Ohnishi, and Tatsuya Harada. 2016. Beyond caption to narrative: Video captioning with multiple sentences. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3364–3368. IEEE.

Alok Singh, Loitongbam Sanayai Meetei, Thoudam Doren Singh, and Sivaji Bandyopadhyay. 2021a. Generation and evaluation of hindi image captions of visual genome. In *Proceedings of the International Conference on Computing and Communication Systems: I3CS 2020, NEHU, Shillong, India*, pages 65–73. Springer.

Alok Singh, Thoudam Doren Singh, and Sivaji Bandyopadhyay. 2020a. A comprehensive review on recent methods and challenges of video description. *arXiv preprint arXiv:2011.14752*.

Alok Singh, Thoudam Doren Singh, and Sivaji Bandyopadhyay. 2020b. Nits-vc system for vatex video captioning challenge 2020. *arXiv preprint arXiv:2006.04058*.

Alok Singh, Thoudam Doren Singh, and Sivaji Bandyopadhyay. 2021b. Attention based video captioning framework for hindi. *Multimedia Systems*, pages 1–13.

Alok Singh, Thoudam Doren Singh, and Sivaji Bandyopadhyay. 2021c. An encoder-decoder based framework for hindi image caption generation. *Multimedia Tools and Applications*, pages 1–20.

Alok Singh, Dalton Meitei Thounaojam, and Saptarshi Chakraborty. 2019. A novel automatic shot boundary detection algorithm: robust to illumination and motion effect. *Signal, Image and Video Processing*, pages 1–9.

Salam Michael Singh, Loitongbam Sanayai Meetei, Thoudam Doren Singh, and Sivaji Bandyopadhyay. 2021d. Multiple captions embellished multilingual multi-modal neural machine translation. In *Proceedings of the First Workshop on Multimodal Machine Translation for Low Resource Languages (MMTLRL 2021)*, pages 2–11, Online (Virtual Mode). INCOMA Ltd.

Ganchao Tan, Daqing Liu, Meng Wang, and Zheng-Jun Zha. 2020. Learning to discretely compose reasoning module networks for video captioning. *arXiv preprint arXiv:2007.09049*.

Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497.

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.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2015. Sequence to sequence-video to text. In *Proceedings of the IEEE international conference on computer vision*, pages 4534–4542.

Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. 2014. Translating videos to natural language using deep recurrent neural networks. *arXiv preprint arXiv:1412.4729*.

Huiyun Wang, Chongyang Gao, and Yahong Han. 2020. Sequence in sequence for video captioning. *Pattern Recognition Letters*, 130:327–334.

Junbo Wang, Wei Wang, Yan Huang, Liang Wang, and Tieniu Tan. 2018. M3: Multimodal memory modelling for video captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7512–7520.

Xin Wang, Jiawei Wu, Junkun Chen, Lei Li, Yuan-Fang Wang, and William Yang Wang. 2019. Vatex: A large-scale, high-quality multilingual dataset for video-and-language research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4581–4591.

Ran Wei, Li Mi, Yaosi Hu, and Zhenzhong Chen. 2020. Exploiting the local temporal information for video captioning. *Journal of Visual Communication and Image Representation*, 67:102751.

Huanhou Xiao, Junwei Xu, and Jinglun Shi. 2020. Exploring diverse and fine-grained caption for video by incorporating convolutional architecture into lstm-based model. *Pattern Recognition Letters*, 129:173–180.

Jun Xu, Tao Mei, Ting Yao, and Yong Rui. 2016. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5288–5296.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR.

Yuecong Xu, Jianfei Yang, and Kezhi Mao. 2019. Semantic-filtered soft-split-aware video captioning with audio-augmented feature. *Neurocomputing*, 357:24–35.

Yang Yang, Jie Zhou, Jiangbo Ai, Yi Bin, Alan Hanjalic, Heng Tao Shen, and Yanli Ji. 2018. Video captioning by adversarial lstm. *IEEE Transactions on Image Processing*, 27(11):5600–5611.

Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. 2015. Describing videos by exploiting temporal structure. In *Proceedings of the IEEE international conference on computer vision*, pages 4507–4515.

Haonan Yu, Jiang Wang, Zhiheng Huang, Yi Yang, and Wei Xu. 2016. Video paragraph captioning using hierarchical recurrent neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4584–4593.

A Scaled Encoder Decoder Network for Image Captioning in Hindi

Santosh Kumar Mishra [1], **Sriparna Saha** [1], **and Pushpak Bhattacharyya** [2]
[1] Department of Computer Science & Engineering, Indian Institute of Technology Patna, India
[2] Department of Computer Science & Engineering, Indian Institute of Technology Bombay, India
{santosh_1821cs03, sriparna}@iitp.ac.in [1], pb@cse.iitb.ac.in [2]

Abstract

Image captioning is a prominent research area in computer vision and natural language processing, which automatically generates natural language descriptions for images. Most of the existing works have focused on developing models for image captioning in the English language. The current paper introduces a novel deep learning architecture based on encoder-decoder with an attention mechanism for image captioning in the Hindi language. For encoder, decoder, and attention, several deep learning-based architectures have been explored. Hindi is the third-most spoken language globally; it is extensively spoken in India and South Asia; it is one of India's official languages. The proposed encoder-decoder architecture employs scaling in convolution neural networks to achieve better accuracy than existing image captioning methods in Hindi. The proposed method's performance is compared with state-of-the-art methods in terms of BLEU scores and manual evaluation. The results show that the proposed method is more effective than existing methods.

1 Introduction

Caption generation from images is a complex job as it necessitates object recognition and articulating the object's relationship in natural language. Caption generation is challenging in comparison to object recognition and image classification, which have been the primary research focus in computer vision. Nowadays, Deep learning-based architecture has emerged as a result of recent developments in machine translation. Recent advances in language modeling, object recognition, and image classification opened up new possibilities. A generated image caption can assist visually challenged individuals to perceive the web content (MacLeod et al., 2017). The architecture based on encoder-decoder has been widely employed to solve the image captioning problem (Karpathy and Fei-Fei, 2015) (Anderson et al., 2018) (Feng et al., 2019). In the literature, two different approaches have been used for caption generation ; the top-down approach (Bahdanau et al., 2014)(Wu et al., 2016) (Vinyals et al., 2015) (Zhou et al., 2020), (Cornia et al., 2020), and the bottom-up approach (Elliott and Keller, 2013)(Kulkarni et al., 2011)(Farhadi et al., 2010).

In this paper, We built a model of caption generation from images in the Hindi language, which is spoken throughout India, South Asia, and other parts of the world as well. It is one of the world's ancient languages and the third most spoken language globally. It originated from the Sanskrit language (Gary and Rubino, 2001). In the literature, there are just a few works on Hindi image captioning (Dhir et al., 2019; Mishra et al., 2021a,b; Singh et al., 2021). The first work was carried out in (Dhir et al., 2019), RESNET 101 (He et al., 2016), and GRU (Cho et al., 2014) is employed in the architecture. In (Mishra et al., 2021a), authors had employed various attention models. In this paper, the authors had explored several architectures with various attention. Authors of (Mishra et al., 2021b) proposed a architecture using transformer. The transformer is employed here as a decoder. This work also utilizes deep-learning based architectures for generating captions of images in the Hindi language. The key contributions of this work are as follows:

- This work is the first of its kind for image captioning in Hindi, which utilizes EfficientNet (Tan and Le, 2019) as an encoder and GRU (gated recurrent unit) (Cho et al., 2014) as a decoder with Bahdanu attention (Bahdanau et al., 2014).

- Ablation study has been conducted with various encoder-decoder and attention technique like X-linear attention (Pan et al., 2020), Bahdanu attention (Bahdanau et al., 2014), , Luong attention (Luong et al., 2015), spatial attention (Lu et al., 2017), Visual Attention (Xu et al., 2015).

- We explored various language model with the proposed architecture like Transformer (Vaswani et al., 2017), LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014).

- We experimented with the newly introduced dataset for image captioning in Hindi (Mishra et al., 2021a) and showed a comparative study between model trained on Hindi dataset and post-processing model. Here, the post-processing model is trained on an English corpus that generates an English caption that is ultimately translated into Hindi. We demonstrated the efficacy of the proposed method by comparing it with the post-processing method.

2 Related Work

In the past, two approaches for image captioning have been used; the first is the top-down approach (Wu et al., 2016)(Sutskever et al., 2014) (Bahdanau et al., 2014), and second approach is an older approach i.e. bottom up approach (Elliott and Keller, 2013)(Kulkarni et al., 2011)(Farhadi et al., 2010). In the first approach, the input image is turned into words, but in the second approach, words define the many features of an image; words are joined to form an image caption. The architecture's parameters are learned in top-down methodology, which comprises end-to-end learning for caption generation.

The language model combines the object characteristics, vocabulary, visual description, and sentences, etc., in the bottom-up approach. Association of the appropriate sentence to an input image is explored in (Farhadi et al., 2010); this sentence is considered as the input image's caption. In (Elliott and Keller, 2013), a template-based method was utilized, it uses visual dependency modeling to record the links among objects.

For image captioning nowadays, the top-down method is very popular. Authors of (Mao et al., 2014) had developed the captioning architecture using the multimodal RNN to generate the caption. A probability distribution model is employed to generate the word based on prior words and an image. The probability distribution is used to generate the image caption. It is analogous to the approach of machine translation employing encoder-decoder architecture. In (Vinyals et al., 2015), authors have used a generative model using an RNN trained to optimize the likelihood of the target sentence given an input image. Authors of (Karpathy and Fei-Fei, 2015) have proposed an image captioning model by using a combination of CNN and RNN over image region utilizing the alignment model. They used bidirectional RNN for language modeling and a structured, objective function aligning two modalities through a multimodal embedding. A language pre-training model unified version is investigated in (Zhou et al., 2020). A meshed memory transformer (Cornia et al., 2020) is utilized for image's feature extraction and language modeling; it learns a multi-level relationship between previous information and regions of the image. Authors of (Liu et al., 2020) have proposed an image captioning model using generative adversarial networks using retrieval and ensemble based approaches. The method given by (Deshpande et al., 2019) has an image captioning structure using variational generative adversarial network and variational autoencoder; the approach generates an image caption based on an image summary. The authors also used part-of-speech as a description that assists in generating the description of the image.

Most of the relevant works for image captioning in the literature are published particularly for the English. Only a limited number of attempts have been made for image caption generation in the Hindi language. In (Dhir et al., 2019), the first attempt for image captioning in the Hindi language is made. A transformer-based image captioning model has been proposed in (Mishra et al., 2021b). In (Mishra et al., 2021a), authors have investigated a variety of architectures with various attention methods for caption generation from images.

3 Proposed Methodology

We employed the encoder-decoder framework with attention for image captioning in the proposed framework (as shown in Fig 1).

3.1 Encoder-Decoder Framework

We explore the encoder-decoder based architecture for caption generation of an image. Given an image, it maximizes the correct description's probability

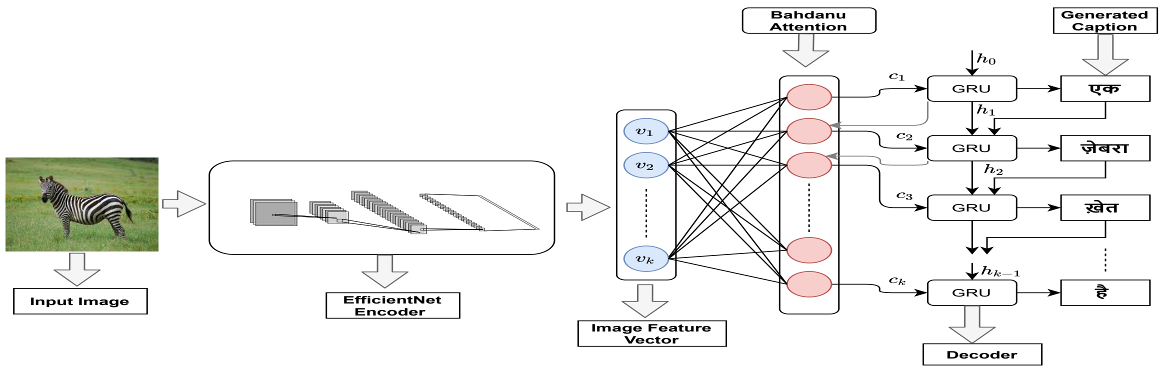


Figure 1: Network architecture of the proposed method

as follows:

$$\theta^* = arg \max_{\theta} \sum_{(I,S)} log\, p(S|I;\theta) \tag{1}$$

In the above equation, θ specifies model parameters, I represents the image and $y = y_1, y_2, ...y_t$ is the related caption. The generated caption y is obtained by chain law. The joint probability distribution's log-likelihood may be calculated as follows:

$$log\, p(y) = \sum_{t=0}^{N} log\, p(y_t|y_0, y_1, ...y_{t-1}, I) \tag{2}$$

For clarity, the model's parameter dependence has been discarded. The architecture based on RNN is as follows:

$$log\, p(y_t|y_0, y_1, ...y_{t-1}, I) = f(h_t, c_t) \tag{3}$$

Here f is a non-linear function finds the next word's output probability. h_t and c_t represents the hidden state and context vector extracted vector of the image at t^{th} time step of recurrent neural network. The context vector c_t is an essential characteristic in this case since it offers verification throughout the caption generating process. (You et al., 2016)(Xu et al., 2015)(Mao et al., 2014)(Vinyals et al., 2015). c_t is dependent on both the encoder and decoder architectures. Its been demonstrated in prior publications; attention helps in increasing the efficiency of the image captioning model (Xu et al., 2015).

3.2 Convolutional Neural Networks as an Encoder for Feature Extraction

The proposed method encodes an input image I into a vector representation of fixed size; the encoded image feature sets the decoder RNN's starting state. We conducted an ablation investigation using encoders such as EfficientNet, Inception V4, and RESNET 101.

3.2.1 EfficientNet

EfficientNet is a group of convolutional neural networks(CNNs) architectures proposed by authors (Tan and Le, 2019) to optimize the accuracy for image classification given a computational cost. It employs the model scaling to find the best combination of resolution, width, and depth in CNNs. There are eight models from B0 to B7 in the EfficientNet, with each subsequent model number relating to variants with more parameters and higher accuracies. We have used the B5 model trained on ImageNet for feature extraction from the input images. More details can be found in the paper (Tan and Le, 2019).

3.2.2 RESNET 101

RESNET101 (Residual Neural Network) (He et al., 2016) is employed for image encoding and extracting features. It consists of 101 layers that are trained on the ImageNet dataset.

3.2.3 Inception V4

This CNN architecture proposed in (Szegedy et al., 2017). It has a greater number of inception components comparing Inception-V3. This is a true Inception variation with no residual links. On ImageNet's test set classification challenge, this model earned a top-5 error of 3.08%.

3.3 Attention Mechanisms

The encoder-decoder model uses a fixed-length context vector which is incapable of remembering long input sequences. The attention mechanisms resolve this problem. Attention mechanisms focus on the crucial part of the image while generating the caption. Proposed encoder-decoder model makes use of, recently introduced X-Linear attention (Pan et al., 2020), Spatial Attention (Lu et al., 2017), Visual Attention (Xu et al., 2015), Luong Attention (Luong et al., 2015), and Bahdanau Attention (Bahdanau et al., 2014).

3.3.1 Bahdanau Attention

This architecture introduced in (Bahdanau et al., 2014) is a well known architecture for sequence to sequence model. This is a kind of additive attention; here context vector is calculated as follows:

$$c_t = \sum_{i=1}^{N} \alpha_{ti} v_i \tag{4}$$

the weight α_{ti} for each feature v_i is determined as :

$$\alpha_{ti} = \frac{exp(e_{ti})}{\sum_{i=1}^{N} exp(e_{ti})} \tag{5}$$

where

$$e_{ti} = f(h_t, v_i) \tag{6}$$

Here, the proposed feed forward neural network (Bahdanau et al., 2014) is denoted by f which is jointly trained on all parameters, v_i denotes image feature, h_t is RNN's hidden state at time step t and N is the generated caption's length.

3.3.2 Luong Attention

This attention mechanism (Luong et al., 2015) is commonly referred as multiplicative attention, which is built upon the Bahdanu attention. Here, c_t denotes the model vector is determined as follows:

$$c_t = \sum_{i=1}^{N} \alpha_{ti} v_i \tag{7}$$

In this case, the weight α_{ik} is determined for each feature v_k as :

$$\alpha_{ti} = \frac{exp(e_{ti})}{\sum_{i=1}^{N} exp(e_{ti})} \tag{8}$$

where

$$e_{ti} = h_t \times w \times v_i \tag{9}$$

e_{ti} represents content based function (Luong et al., 2015), v_i is the image feature vector, N is the length of the generated caption, h_t denotes hidden states at time step t, and w represents the learnable parameters.

3.3.3 Visual Attention

Authors of (Xu et al., 2015) demonstrated an attention technique for focusing on the appropriate portion of the image while generating a caption. Here, the context vector is computed using:

$$e_{ti} = f(v_i, h_t) \tag{10}$$

$$\alpha_{ti} = \frac{exp(e_{ti})}{\sum_{i=1}^{N} exp(e_{ti})} \tag{11}$$

$$c_t = \phi(v_i, \alpha_{ti}) \tag{12}$$

Here, d dimensional feature vectors of different parts of the images are $V = [v_1, v_2........v_k]$, $v_i \epsilon R^d$ is the spatial image feature. h_t denotes the hidden state of recurrent neural network at t^{th} time step. α_{ti} denotes the weight which is computed for each image feature vector, v_i, at every time step by a proposed attention architecture, f (Xu et al., 2015). It employs a multilevel perceptron applied on hidden state, h_t, and context vector, c_t. The function ϕ returns a single vector corresponding to their weights, further h_t and c_t are jointly utilized to anticipate the succeeding word as given in Equation 3.

3.3.4 Spatial Attention

This is generated from the residual network (He et al., 2016). This mechanism utilizes residual connection (He et al., 2016); authors have introduced a new technique of determining the context vector, it is regarded as the present hidden state's residual visual information.

$$c_t = g(V, h_t) \tag{13}$$

Here attention function is represented by g and $V = [v_1, v_2........v_k]$, $v_i \epsilon R^d$ represents d dimensional feature vector of image. v_i and h_t represent parts of image and RNN's hidden state at time step t, respectively.

3.3.5 X-Linear Attention

The conventional attention module primarily uses first-order interaction for image captioning, which has limited multi-modal reasoning capacity. Second-order interaction (bilinear pooling) has been demonstrated to be helpful in visual recognition by the authors of (Gao et al., 2016), and (Yu et al., 2018). This mechanism has been utilized by the authors of (Kim et al., 2018), and (Fukui et al., 2016) for visual question answering. X-linear attention uses bilinear pooling that boosts the attended feature's capacity of representation by utilizing the higher-order interaction between uni-modal and multi-modal features.

Let's suppose that $Q \in R^{D_q}$ represent the query, $K = \{k_i\}_{i=1}^{N}$ represent the keys and $V = \{v_i\}_{i=1}^{N}$ denote the set of values, where $v_i \in R^{D_v}$ and $k \in R^{D_k}$ are i^{th} value and key pair, respectively. Lower rank bi-linear pooling is used by X-linear attention to obtain a query-key representation, $B_i^k \in R^{D_B}$, between query, Q, and each key, k_i.

$$B_i^k = \sigma(W_k k_i \odot \sigma(W_q^k Q)) \tag{14}$$

$W_q^k Q$ and $W_k \in R^{D_B \times D_k}$ are the embedding matrices, sigma (σ) depicts the relu activation function and $\odot$ is the multiplication of elements. Here B_i^k specifies learned bi-linear query representation, and it represents an interaction between the key and query on a second-order level.

Furthermore, two types of bi-linear distributions of attention are calculated to aggregate both channel-wise and spatial information across all values. Two embedding layers are used to get the distribution of spatial attention. The bi-linear query representation is then projected into corresponding attention weight using a softmax layer.

$$B_i^{'k} = \sigma(W_B^k B_i^k) \tag{15}$$

$$b_i^s = W_b B_i^{'k} \tag{16}$$

$$\beta^s = softmax(b^s) \tag{17}$$

Where W_b and $W_B^k \in R^{D_c \times D_B}$ are the embedding matrices. $B_i^{'k}$ represents bi-linear query-key representation and b_i^s denotes the i^{th} element in b^s. Each of the elements β_i^s in β^s represents a key/value pair's attention weight. Further squeeze-excitation (Hu et al., 2018) is performed over all transformed bi-linear query representations, $\{B_i^{'k}\}_{i=1}^{N}$, for attention measurement in channel wise manner. The squeeze operation uses average pooling to aggregate all of the modified bi-linear key and query representations, yielding a global channel descriptor, $\overline{B}$, as follows:

$$\overline{B} = \frac{1}{N}\sum_{i=1}^{N} B_i^{'k} \tag{18}$$

Further, channel wise attention distribution is derived by excitation operation, β^c, by using the self gating with sigmoid activation function over the global channel descriptors, $\overline{B}$.

$$b^c = W_e \overline{B} \tag{19}$$

$$\beta^c = sigmoid(b^c) \tag{20}$$

The embedding matrix is $W_e \in R^{D_B \times D_c}$ in this case. Finally, the X-Linear attention module produces the attended features of images by combining improved bi-linear values with channel-wise and spatial attention to form the attended feature.

$$\hat{v} = F_{X-linear}(K, V, Q) = \beta^c \odot \sum_{i=1}^{N} \beta_i^s \beta_i^v \tag{21}$$

$$B_i^v = \sigma(W_v v_i) \odot \sigma(W_q^v Q) \tag{22}$$

Where $W_v \in R^{D_B \times D_v}$ and $W_q^v \in R^{D_B \times D_q}$ are the embedding matrices, B_i^v represents the bi-linear pooling's enhanced values on query, Q, and value, v_i. In contrast to the traditional attention framework that utilizes only the first-order interaction, the X-linear attention model utilizes the second-order interaction via bi-linear pooling. Therefore, it has more representative attended features than the traditional attention framework.

3.4 Decoder for Language Modeling

We have used various decoder models for ablation study and to determine the best possible architecture. The language modeling RNN has the challenge of exploding and vanishing gradient (Hochreiter and Schmidhuber, 1997). This problem can be resolved employing gated recurrent unit (Cho et al., 2014), and Long Short-Term Memory (Hochreiter and Schmidhuber, 1997). We have included a bi-directional variation in addition to the uni-directional GRU and LSTM, which enables the networks to have forward and backward sequence information. We have also incorporated the transformer (Vaswani et al., 2017) as a decoder; apart from attention, to enable optimization easier and quicker, it employs positional encoding, residual connection, and layer normalization.

4 Experimental Setup

This section covers the methods employed to create the dataset and evaluate the proposed methodology.

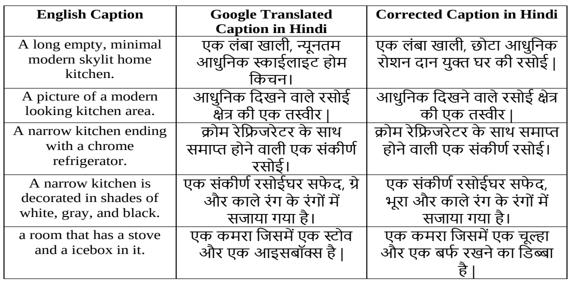
4.1 Dataset

The authors of (Mishra et al., 2021a) generated the Hindi variant of the MSCOCO dataset. This is a popular dataset for caption generation from images. In this dataset, each image has five captions. There are 82573, 811, 811 images for training, testing, and validation. The captions in the training set, validation set, and test set are about 4 lakh, 4000, and 4000, respectively. Despite the fact Google Translate is employed for the translation, the following difficulties have been experienced when translating from English to Hindi:

- Because Google Translate lacks a system to assess the context of the statement, the context of the translated caption is lost during translation.
- In certain cases, Google Translator's translation is grammatically imprecise.
- Google Translator's accuracy is not standardized because it depends on the source and target languages.

Therefore, human annotators are employed to correct Google translated sentences to remove errors. The inter-annotator agreement was 87% between two annotators. Figures 2 and 3 display a sample from the dataset that was created.

Figure 2: Example Image for Dataset Preparation



English Caption	Google Translated Caption in Hindi	Corrected Caption in Hindi
A long empty, minimal modern skylit home kitchen.	एक लंबा खाली, न्यूनतम आधुनिक स्काईलाइट होम किचन।	एक लंबा खाली, छोटा आधुनिक रोशन दान युक्त घर की रसोई |
A picture of a modern looking kitchen area.	आधुनिक दिखने वाले रसोई क्षेत्र की एक तस्वीर |	आधुनिक दिखने वाले रसोई क्षेत्र की एक तस्वीर |
A narrow kitchen ending with a chrome refrigerator.	क्रोम रेफ्रिजरेटर के साथ समाप्त होने वाली एक संकीर्ण रसोई।	क्रोम रेफ्रिजरेटर के साथ समाप्त होने वाली एक संकीर्ण रसोई।
A narrow kitchen is decorated in shades of white, gray, and black.	एक संकीर्ण रसोईघर सफेद, ग्रे और काले रंग के रंगों में सजाया गया है।	एक संकीर्ण रसोईघर सफेद, भूरा और काले रंग के रंगों में सजाया गया है।
a room that has a stove and a icebox in it.	एक कमरा जिसमें एक स्टोव और एक आइसबॉक्स है |	एक कमरा जिसमें एक चूल्हा और एक बर्फ रखने का डिब्बा है |

Figure 3: Example of Dataset Preparation

4.2 Evaluation Metric

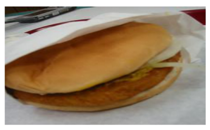
We employed the BLEU score (Papineni et al., 2002), a standard evaluation measure used in image captioning and machine translation etc.

4.3 Hyperparameters Used

EfficientNet extracts feature from $224 * 224$ input images and transform them into $49 * 512$ feature vectors. Embedding layer size is 512 neurons, 0.4 dropouts are employed to prevent over-fitting. The batch size is fixed to 128, and the epochs are set to 15. Softmax cross-entropy is employed as a loss function. The Adam optimizer with a $4e-4$ learning rate is used for optimization. It takes 14 hours to train; a caption for the image needs around 30 to 40 seconds to generate.

Methods	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Proposed Methodology (EN + BA + GRU)	**67.3**	**48.5**	**33.1**	**22.0**
EN + BA + LSTM	66.7	47.6	32.4	21.8
EN + VA + GRU	66.5	47.3	32.1	21.4
EN + XLA + GRU	66.8	47.8	32.1	21.0
EN + XLA + LSTM	67.2	48.0	32.7	21.9
EN + XLA + Bi-GRU	66.1	47.0	31.5	20.6
EN + XLA + Bi-LSTM	66.7	47.6	32.1	21.2
IV4 + SA + GRU	55.7	38.4	25.9	16.8
EN + Trans	62.7	43.7	28.8	18.2
RN101 + SA + LSTM	56.4	38.8	26.3	17.2
IV4 + SA + LSTM	56.3	38.4	25.8	16.9
EN + LA + GRU	67.0	48.4	32.4	21.0
EN + SA + GRU	66.0	46.7	31.4	20.5
Mishra et al.(Mishra et al., 2021a)	67.0	47.8	31.9	21.2
Mishra et al. (Mishra et al., 2021b)	62.9	43.3	29.1	19.0
Dhir et al. (Dhir et al., 2019)	57.0	39.1	26.4	17.3

Table 1: The score obtained with various architectures and comparison with existing methods. Here Trans, Bi-LSTM, LSTM, Bi-GRU, GRU, SA, LA, BA, VA, XLA, IV4, RN101, and EN represents Transformer, Bi-directional Long Short-Term Memory, Long Short-Term Memory, Bi-directional Gated Recurrent Unit, Gated Recurrent Unit, Spatial Attention, Luong Attention, Bahdanu Attention, Visual Attention, X-Linear Attention, Inception V4, RESNET101, and EfficientNet



C1: एक सफेद प्लेट पर एक हॉट डॉग
C2: A hot dog on a table
C3: एक मेज पर एक गर्म कुत्ता

C1: Generated caption based on model trained on Hindi dataset
C2: Generated caption based on model trained on English dataset
C3: Generated caption based on model trained on English dataset then translated into Hindi

Figure 4: Captions generated by different models of test images. Generated caption, Gloss and Transliteration are denoted by I,II, and III.

5 Results and Discussions

A comprehensive overview of obtained results and generated captions are discussed in this section.

Score	Adequacy (Meaning)	Fluency(Meaning)
0	Poor: In the caption generated, none of the information is retained.	Poor: The Hindi caption generated is incomprehensible.
1	Bad: There is little information retained in the caption generated.	Bad: The Hindi caption generated is dis-fluent.
2	Moderate: Much of the information in the caption generated are retained.	Moderate: The Hindi captions generated are like non-native Hindi captions.
3	Good: Most of the information in the caption produced is retained.	Good: In terms of Hindi grammar rules, the generated Hindi captions are good.
4	Excellent: In the produced caption, all of the information are retained.	Excellent: Hindi captions generated are correct in terms of Hindi grammar rules.

Table 2: Adequacy and fluency measurement scale

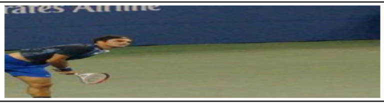
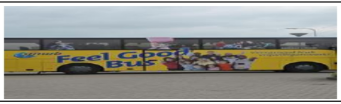
(a): I- एक बिल्ली एक सोफे पर लेटी हुई है II- One cat one couch on lying down III- Ek bille ek sofe par leti hui hai	(b): I- एक आदमी एक सेल फोन पर बात कर रहा है II- One man one cell phone on talking III- Ek adami ek cell phone par bat kar raha hai	(c): I- एक मेज पर फलों और सब्जियों का एक गुच्छा II- One table on fruits and vegetables of one bunch III- Ek mej par phalo aur sabjiyo ka ek guchha
		
(d): I- एक आदमी एक टेनिस कोर्ट पर टेनिस खेल रहा है II- One man one tennis court on tennis playing III- Ek adami ek tennis court par tennis khel raha hai	(e): I- हाथियों का एक समूह एक खेत में घूम रहा है II- Elephant of one group one field in moving III- Hathiyo ka ek samooh ek khet me ghoom raha hai	(f): I एक पीले रंग की बस जो सड़क पर खड़ी है II- One yellow color of bus which road on parked III- Ek peele rang ke bus jo sadak par khadee hai

Figure 5: Generated qualitative results on test images. Generated caption, Gloss and Transliteration are denoted by I,II, and III.

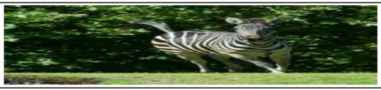
		
(a): I-एक ज़ेबरा एक खेत में खडा है II- One zebra one field in standing III- Ek zebra ek khet me khada hai	(b): I- घास में एक मैदान में खड़े मवेशियों का झुंड II- Grass in one ground in standing of herd of cattle III- Ghas me ek maidan me khade maveshiyon ka ek jhund	(c): I- एक झील के ऊपर एक बड़ा सफेद नाव पानी में है II- One lake of above one big white boat in water III- Ek jheel ke upar ek bada safed naav pani me hai

Figure 6: Qualitative results to show error analysis on test images. Generated caption, Gloss and Transliteration are denoted by I,II, and III.

5.1 Comparisons with existing methods for Image Captioning in Hindi

The following works have been undertaken for image captioning in Hindi as per our understanding:

- In (Dhir et al., 2019), author have proposed the architecture for caption generation, where they had used RESNET 101 (He et al., 2016) and GRU (Cho et al., 2014).
- A transformer-based architecture introduced in (Mishra et al., 2021b), where transformer is utilized for language modeling.
- (Mishra et al., 2021a) investigates a variety of architectures with various attention methods for Hindi image captioning.

As a result, we evaluated our technique to these approaches, Table 1 show that our approach beats the existing method and baselines of the ablation study.

5.2 Qualitative Analysis

We cover the qualitative examination of our approach using test images in this section. The captions for the test images that were generated are shown in Fig 5. Gloss annotations and transliterations are added for non-Hindi speakers; they help comprehend the captions in Hindi. It is obvious that the produced captions are mostly accurate and can appropriately signify the items and activities depicted in the images.

5.3 Quantitative Analysis

The efficiency of the proposed method was assessed using BLEU scores, as can be seen in Table 1. This table depicts that our method surpasses existing approaches considering BLEU. This demonstrates the effectiveness of our approach.

5.3.1 Human Evaluation Based on Adequacy and Fluency

These metrics are widely employed in various natural language processing problems, for-instance summarization, question-answering, and machine translation. Adequacy measures information retained in the caption generated and fluency tests generated caption in terms of grammatical norms. These metrics were evaluated on a scale of 0 to 4 (as indicated in Table 2). Two human annotators have accomplished this task with an agreement of 87% between them.

The generated captions of two approaches have been measured here:

- The approach employs a dataset including Hindi corpora in the training phase. The trained model generates captions in Hindi. This yields a score of 3.112 for adequacy and 3.233 for fluency.

- Another approach uses an English corpus for training and generates captions in English. The Google Translator is being used to convert the produced English caption into Hindi. This yields a score of 2.142 for adequacy and 2.761 for fluency.

Our methodology is superior to the post-processing procedure (The Hindi captions are formed by translating the English captions generated by the trained model with the English corpus.). The generated captions are presented in Fig 4, and the result is that the model trained on the Hindi dataset beats the post-processing procedure, which highlights the need for a Hindi dataset.

5.4 Error Analysis

There are some challenges for the image captioning framework that results in errors during caption generation (as shown in Fig 6). These challenges could be categorized as following:

- **Recognition of activity**: As can be observed in Fig 6 (a), a zebra is really sprinting, yet the model predicted that it would be standing.' This might be because the bulk of the images in dataset has a standing zebra.
- **Objects counting:** There are two animals in the picture in 6 (b), however, the model predicted 'herd of cattle.' This might be due to trained CNN's inability to detect the number of objects.
- **Occlusion:** It occurs when objects are partially visible or so near that the machine learning model can't recognize them. As can be observed in Fig 6 (c), In the caption, the model predicted 'boat' rather than 'aeroplane.'

6 Conclusion and Future Work

We present a novel approach for caption generation from images in Hindi that employs an encoder-decoder model based on EfficientNet and GRU, as well as attention techniques. We use EffcientNet as an encoder because its efficacy outperforms state-of-the-art CNNs for image classification and feature extraction. We use a gated recurrent unit as a decoder for language modeling as it is less computationally expensive and it achieves state-of-the-art efficacy for language modeling. Further, the use of Bahdanau attention makes the system robust. Aside from that, we undertake an ablation analysis to find the ideal architecture. The proposed methodology could be expanded for image-to-paragraph generation and dense image captioning.

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

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

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, 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*.

Marcella Cornia, Matteo Stefanini, Lorenzo Baraldi, and Rita Cucchiara. 2020. Meshed-memory transformer for image captioning. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Aditya Deshpande, Jyoti Aneja, Liwei Wang, Alexander G. Schwing, and David Forsyth. 2019. Fast, diverse and accurate image captioning guided by part-of-speech. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Rijul Dhir, Santosh Kumar Mishra, Sriparna Saha, and Pushpak Bhattacharyya. 2019. A deep attention based framework for image caption generation in hindi language. *Computación y Sistemas*, 23(3).

Desmond Elliott and Frank Keller. 2013. Image description using visual dependency representations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1292–1302.

Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: Generating sentences from images. In *European conference on computer vision*, pages 15–29. Springer.

Yang Feng, Lin Ma, Wei Liu, and Jiebo Luo. 2019. Unsupervised image captioning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv preprint arXiv:1606.01847*.

Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. 2016. Compact bilinear pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 317–326.

Jane Gary and Carl Rubino. 2001. Facts about the world's languages: An encyclopedia of the world's major languages.

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.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-excitation networks, 7132–7141. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Salt Lake City, UT*.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137.

Jin-Hwa Kim, Jaehyun Jun, and Byoung-Tak Zhang. 2018. Bilinear attention networks. In *Advances in Neural Information Processing Systems*, pages 1564–1574.

Girish Kulkarni, Visruth Premraj, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C Berg, and Tamara L Berg. 2011. Baby talk: Understanding and generating image descriptions. In *Proceedings of the 24th CVPR*. Citeseer.

Junhao Liu, Kai Wang, Chunpu Xu, Zhou Zhao, Ruifeng Xu, Ying Shen, and Min Yang. 2020. Interactive dual generative adversarial networks for image captioning. In *AAAI*, pages 11588–11595.

Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2017. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 375–383.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Haley MacLeod, Cynthia L Bennett, Meredith Ringel Morris, and Edward Cutrell. 2017. Understanding blind people's experiences with computer-generated captions of social media images. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 5988–5999. ACM.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. 2014. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*.

Santosh Kumar Mishra, Rijul Dhir, Sriparna Saha, and Pushpak Bhattacharyya. 2021a. A hindi image caption generation framework using deep learning. *Transactions on Asian and Low-Resource Language Information Processing*, 20(2):1–19.

Santosh Kumar Mishra, Rijul Dhir, Sriparna Saha, Pushpak Bhattacharyya, and Amit Kumar Singh. 2021b. Image captioning in hindi language using transformer networks. *Computers & Electrical Engineering*, 92:107114.

Yingwei Pan, Ting Yao, Yehao Li, and Tao Mei. 2020. X-linear attention networks for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10971–10980.

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.

Alok Singh, Thoudam Doren Singh, and Sivaji Bandyopadhyay. 2021. An encoder-decoder based framework for hindi image caption generation. *Multimedia Tools and Applications*, pages 1–20.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information systems*, pages 3104–3112.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Mingxing Tan and Quoc V Le. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*.

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.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164.

Qi Wu, Chunhua Shen, Lingqiao Liu, Anthony Dick, and Anton Van Den Hengel. 2016. What value do explicit high level concepts have in vision to language problems? In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 203–212.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4651–4659.

Chaojian Yu, Xinyi Zhao, Qi Zheng, Peng Zhang, and Xinge You. 2018. Hierarchical bilinear pooling for fine-grained visual recognition. In *Proceedings of the European conference on computer vision (ECCV)*, pages 574–589.

Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason J Corso, and Jianfeng Gao. 2020. Unified vision-language pre-training for image captioning and vqa. In *AAAI*, pages 13041–13049.

Co-attention based Multimodal Factorized Bilinear Pooling for Internet Memes Analysis

Gitanjali Kumari[1] **Amitava Das**[2] **Asif Ekbal**[1]
[1]Department of Computer Science and Engineering,
[1]Indian Institute of Technology Patna, India
[2]Wipro AI Labs, Bangalore, India
{gitanjali_2021cs03,asif}@iitp.ac.in
amitava.das2@wipro.com

Abstract

Social media platforms like Facebook, Twitter, and Instagram have a significant impact on several aspects of society. Memes are a new type of social media communication found on social platforms. Even though memes are primarily used to distribute humorous content, certain memes propagate hate speech through dark humor. It is critical to properly analyze and filter out these toxic memes from social media. But the presence of sarcasm and humor in an implicit way makes analyzing memes more challenging. This paper proposes an end-to-end neural network architecture that learns the complex association between text and image of a meme. For this purpose, we use a recent SemEval-2020 Task-8 multimodal dataset. We proposed an end-to-end CNN-based deep neural network architecture with two sub-modules viz.(i) Coattention based submodule and (ii) Multimodal Factorized Bilinear Pooling(MFB) submodule to represent the textual and visual features of a meme in a more fine-grained way. We demonstrated the effectiveness of our proposed work through extensive experiments. The experimental results show that our proposed model achieves a 36.81% macro F1-score, outperforming all the baseline models.

1 Introduction

Social media such as Facebook, Twitter, Instagram, etc., are interactive platforms that accelerate the idea of creating and sharing information. This information made an enormous impact in different fields of society more powerfully and effectively. But on the other hand, we observe a significantly large amount of offensive content in the various social networking sites, which spread hatred, rumors, etc., between the different communities, groups, or individuals. Meme (Dawkins, 2016) is the form of multimodal media that has been initially created to spread humorous content, but due to its multimodal nature, some memes help users to spread hate speech in the form of dark humor. On social media, posting such memes to troll, cyberbully, or targeting someone is increasing rapidly. Unlike other multimodal tasks (e.g., Visual Question Answering, Image Captioning, etc.), in sentiment analysis for memes, textual and visual information are very weakly semantically aligned. In such a situation, we cannot uncover the complex meaning of hateful content until we get to know both the modalities and their contributions in any content hateful. Analysis of such memes can bring valuable insights that are not explored yet. For example, if there is a meme with text containing "Look how many people love you." The sentiment of this meme can, itself, be positive, negative, or neutral. The sentiment can only be found if and only if we add an image to it (c.f. Figure 1). Some memes are purely humorous, while others spread offensive content in the form of dark humor, sarcasm, mockery, etc. Sentiment analysis of memes in a more effective way will facilitate combating such social media issues.

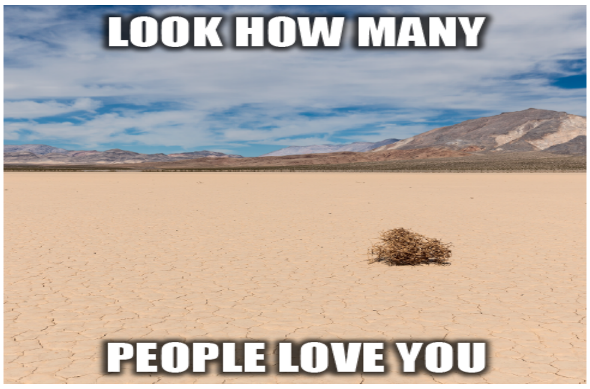


Figure 1: A meme where only after focusing on both text and image, negative sentiment can be identified.

With the phenomenal growth of social media networks, sentiment analysis plays a significant role in handling various aspects of political and religious views of society. Sentiment analysis research has been a progressive area in Natural Language Processing (NLP). It is ranging from document-level classification (Lin and He, 2009; Mouthami et al., 2013) to learning the word and phrase polarity(Hatzivassiloglou and McKeown, 1997; Esuli and Sebastiani, 2006). Several supervised machine learning and feature-based techniques have been used to tackle this problem (Lai et al., 2015; Kouloumpis et al., 2011). However, deep learning-based techniques have gained a lot of popularity in recent years (Truong and Lauw, 2019). A significant amount of works have been done which includes the analysis of opinions about hotel reviews ((Kasper and Vela, 2011; Shi and Li, 2011)), product reviews ((Cernian et al., 2015; Wei and Gulla, 2010; Fang and Zhan, 2015)) etc. Initially, sentiment analysis has been carried out mostly using text (Badjatiya et al., 2017; Davidson et al., 2017; Fortuna et al., 2019). Recently due to the growth of multimedia contents in social media, we also need to develop robust models that would deal with multimodal content. There have been very few attempts towards analyzing the sentiment of memes by researchers. However, research shows that there is a far way to go if we compare the system-generated output to the human evaluation. There can be several reasons for this, such as hateful meaning hidden behind humor, sarcasm, the use of very twisted words, or image to spread hate (Sharma et al., 2020). Lack of annotated datasets can also be one of the reasons for this kind of failure.

The key attributes of our current work can be summarized as follows: (i) We develop a deep neural network-based architecture to explore the idea of co-attention to identify the impact of text and image simultaneously for predicting the correct sentiment of a given meme.(ii) We also explore the concept of Multimodal Factorized Bilinear pooling to represent the textual and visual features in a internet meme analysis dataset. We test the significance of our proposed method on SemEval2020 (Sharma et al., 2020) dataset. Evaluation results the accuracy and macro-F1 of 54% and 36.81%, respectively, which are higher than the baseline model for the given task.

2 Related Work

This section briefly discusses the review related to two aspects: $a)$ Sentiment analysis for unimodal data, $b)$ Sentiment analysis for multimodal data.

2.1 Sentiment analysis in unimodal data

With the emergence of social media and vast internet content, Sentiment analysis has received much attention in the Natural Language Processing (NLP) community. It is very useful on topical categorization task to sort documents according to their subjects such as economics or politics ((Ali et al., 2019; Ilyas et al., 2020)). The work reported in (Kouloumpis et al., 2011) investigated the usefulness of some linguistic features and other features to get an idea about the informal and creative language used in microblogging. Similarly, authors in (Agarwal et al., 2011) proposed to use Part-of-Speech (PoS)-specific prior polarity features to examine sentiment analysis on Twitter data. (Hu and Flaxman, 2018) pioneered HEMOS (Humor-EMOji-Slang-based), a kind of fine-grained sentiment analysis system for the Chinese language to investigate the significance of perceiving the impact of humor, pictograms, and slang to affect users on social media. To address the challenge of sentiment reflection prediction in visual content,(Borth et al., 2013) initiated a data-driven systematic approach by using psychology theories to construct a Visual Sentiment Ontology(VSO) which is a collection of 3,000 Adjective Noun Pairs (ANP) to construct SentiBank, a mid-level concept representation of each image to characterize the sentiment reflected in any visual content. Similarly, we also see a few works on aggression detection from the given textual data (Kumar et al., 2018; Xu et al., 2012).

2.2 Sentiment analysis in multimodal data

Although multimedia content is significantly growing on social media, it is a great challenge to uncover the underlying sentiment mentioned in these. Multimodal sentiment analysis for detecting the polarity of image and text is similar to finding out the hateful content in internet memes. It is also observed that deep learning techniques significantly outperform when it is compared to the traditional machine learning approaches on multimodal data (Kumar et al., 2020; Tran and Cambria, 2018). VistaNet (Lecun et al., 2015) shows the significant importance of visual knowledge in the visual and

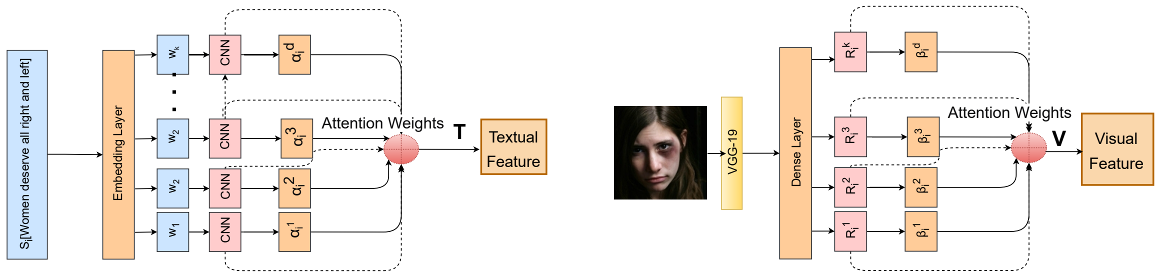


Figure 2: Textual and image feature after applying self-attention

textual content for detecting a sentiment of a document. The research reported in (Yang et al., 2019) tried to explore several deep learning techniques to integrate textual and visual parts of a meme.

The authors in (Yu et al., 2019) pioneered a cascade of Modular Co-Attention Network (MCAN) with a cascade of modular co-attention (MCA) layers, each of which consists of the self-attention and guided-attention units to model the intra and inter modal interactions synergistically. Furthermore, a Dynamic Co-attention Network(DCN) for VQA was introduced in (Xiong et al., 2018). In a paper, (Sabat et al., 2019) reported how visual modality can bring more information than linguistic information for the hateful memes classification task. A hierarchical method for multimodal processing of features using deep learning techniques has shown good result(Majumder et al., 2018). Authors in a paper (?) introduced the idea of multimodal factorized bilinear pooling with co-attention to demonstrate that MFB with co-attention on the real-world VQA dataset achieves new state-of-the-art performance.

Based on the above literature survey, we understood the need to develop such a robust model that can quickly identify the sentiment of a given meme. In our work, we explored the significance of the co-attention and MFB mechanism on the multimodal sentiment analysis task.

3 Methodology

Our current task aims at determining the sentiment of a given meme in a multimodal dataset. The problem can be defined as follows: Given every meme M_i in the dataset which is a combination of text $T_i = (t_{i1}, t_{i2}, ..., t_{ik})$ and image I_i with the shape (224,224,3) in RGB pattern, our task is to create one classifier that should predict one correct label $Y \subseteq$\{neg,neu,pos\} for M_i i.e. predict the correct sentiment whether a given meme is negative, neutral or positive. The respective optimizing goal is then to learn the parameter θ and get the optimum loss function $L(Y|M, \theta)$.

At first, we develop a unimodal baseline system for text and image each. Finally, different multimodal approaches for the fusion of both modalities, i.e., textual and visual, have been described in the following sections:

3.1 Embedding Layer

At first, the pre-processing is performed on the texts, which include stop-word removal and lowercasing of tokens.After pre-processing, every word of each sentence $S_i = (w_{i1}, w_{i2}, ..., w_{ik})$ where k is the max_length of the sentence, is represented using its semantic representation. Each word w_{ij} is transformed into a pre-defined size of the vector, which contains the semantic meaning of that word, known as the word embedding vector. In our experiment, we use FastText (Bojanowski et al., 2017) word embedding for the same purpose. This embedding vector is passed through a deep neural network-based classifier for the sentiment analysis task further.

3.2 Textual Features

We use the convolutional neural network(CNN) (Simonyan and Zisserman, 2015) architecture for identifying the sentiment of a given textual part of the meme. The CNN model consists of three layers, namely convolutional, pooling, and fully connected layers. Textual features are extracted from the fully connected layer. For our experiment, we use three convolution layers with filter sizes 2, 3, and 4. Each convolution layer consists of 128 filters. Equation 1 shows the textual feature vector T_i of a sentence S_i after passing it through

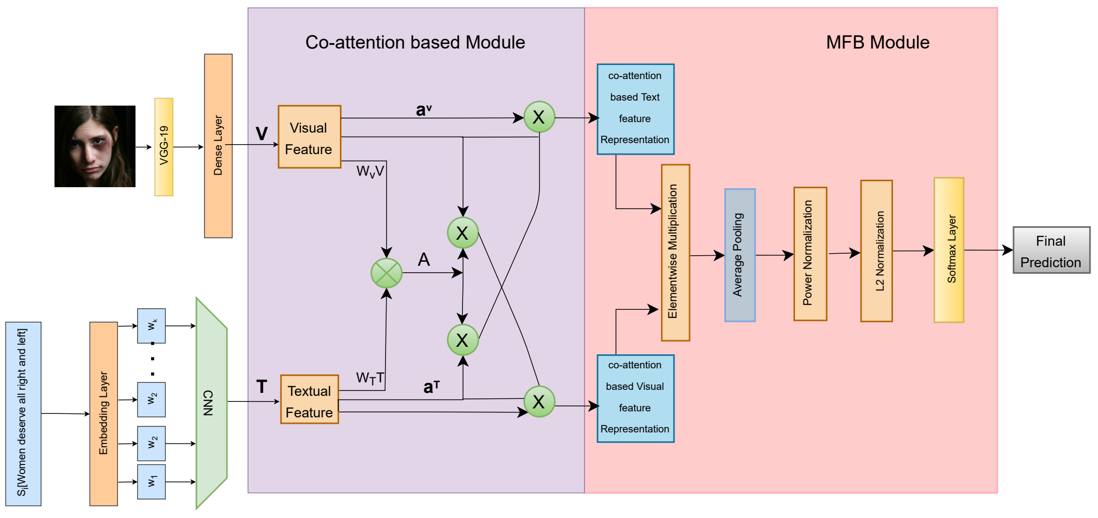


Figure 3: Our proposed co-attention with Multimodal Factorrized Bilinear Pooling based Model

convolution neural network.

$$T_i = (t_i^1, t_i^2, ...t_i^d) \tag{1}$$

3.3 Textual Features with self-attention

On top of this textual feature, we use the attention mechanism. For a given sentence S, the attention model finds out the most important words with the help of attention weights, which is beneficial for the decision-making purposes.

The text feature after applying attention is the weighted sum of all the words present in the sentence S_i. It uses attention weights of each source word, as given in equation 2.

$$c_i^k = \sum_{j=1}^{d} \alpha_{ij} h_j \tag{2}$$

We find out the attention score α_i^j for every feature representation w_{ij} of each word t_i^j in the sentence S_i which is given in equation 3.

$$\alpha_i^j = \frac{exp(e_i^j)}{\sum_{j=1}^{d} exp(e_i^j)} \tag{3}$$

where,

$$e_i^j = \theta(W t_i^j + b) \tag{4}$$

3.4 Visual Features

For extracting the visual features, we use the pre-trained VGG-19 model, which is trained on Imagenet (Simonyan and Zisserman, 2015) dataset. Image with the input shape (224*224) is given to the VGG19 architecture. In VGG19, we kept all the lower layers frozen and extracted the output of the $block5 - conv4$ layer. The extracted output has 196 regions, and 512 dimensions represent each region. So, finally, we obtain a region feature with (196*512) dimensions passed to one dense layer with 250 neurons.

3.5 Visual Features with self-attention

The output from the dense layer mentioned in Section 3.4 is passed through the attention layer to obtain the most important regions that play a vital role in image classification.

Equation 5 shows the region feature of an image I_i

$$R_i = (R_i^1, R_i^2, ..., R_i^k) \tag{5}$$

We apply attention on top of the textual features in Section (3.2). Similarly, we use attention on the top of the region feature R_i of an image. After attention, the visual feature is the weighted sum of all the regions present in the Image V_i. It uses attention weights of each region, as given in equation 6.

$$c_i^k = \sum_{j=1}^{d} \beta_{ij} R_j \tag{6}$$

We find out the attention score β_i^j for every region R_i^j in R_i which is given in equation 7.

$$\beta_i^j = \frac{exp(h_i^j)}{\sum_{j=1}^{d} exp(h_i^j)} \tag{7}$$

where,

$$h_i^j = \theta(W R_i^j + b) \tag{8}$$

Table 1: Result of models

Model	Modality	Fusion	F1-Score	Accuracy
Model 1	SemEval2020 baseline	-	0.2176	–
Model 2	Only Text	-	0.3278	0.40
Model 3	Text+Self-Attention	-	0.3397	0.49
Model 4	Only Image	-	0.2936	0.34
Model 5	Image+Self-Attention	-	0.3166	0.37
Model 6	Text+ Image	early fusion with concatenation	0.3222	0.51
Model 7	Proposed Model1	Co-attention model Bilinear Pooling	**0.3532**	**0.52**
Model 8	Proposed Model2	Co-attention with MFB	**0.3681**	**0.54**
(Keswani et al., 2020)	SemEval2020(SOTA)	-	0.3546	–

$$\sum_{t=1}^{T} P(\hat{y}_0(x,t)|x).s(\hat{y}_0(x,t)) \tag{9}$$

3.6 Fusion of textual and visual features for baseline model

After extracting the textual and visual features separately, we use a fusion technique for our baseline model where we merely concatenate both textual T_i and visual feature V_i. This concatenated feature vector is passed through one dense layer which follows one softmax layer. The softmax layer gives the probability distribution for each class to classify the given meme into pre-defined categories.

3.7 Proposed model

Co-attention: Along with self-attention for textual and visual features, we also explore the concept of co-attention to introduce a natural symmetry between text and image where image representations guide the textual attention and textual representation guide the visual representation(Lu et al., 2016).

For a given textual feature T $\in$R$^{(d\times T)}$ and given a visual feature T $\in$R$^{(d\times V)}$, we calculate a similarity matrix representation called as affinity matrix A $\in$R$^{(T\times V)}$ as follows:

$$A = \tanh{(T^T W_b V)} \tag{10}$$

Using the affinity matrix A in equation 10, we calculate the textual and visual attention maps in the following way:

$$\begin{aligned} H_V &= \tanh{((W_t T)A + W_v V)} \\ a^V &= \mathit{softmax}(w_{hv}^T H_V) \end{aligned} \tag{11}$$

$$\begin{aligned} H_T &= \tanh{((W_t T + (W_v V)A_T)} \\ a^T &= \mathit{softmax}(w_{ht}^T H_T) \end{aligned} \tag{12}$$

Here, W_t, W_v $\in$R$^{(k\times d)}$ and w$_{ht}^T, w_{hv}^T$ are weight matrix. a^V and a^T are the attention probabilities of image and textual part, respectively.

After that, we calculate the textual (T_V) and visual (I_V) attention vector, which is the weighted sum of textual and visual features.

$$T_V = \sum_{t=1}^{T} a_i^T T_i \tag{13}$$

$$I_V = \sum_{i=1}^{N} a_i^V V_i \tag{14}$$

3.7.1 Fusion of textual and visual features with bilinear Pooling

In the earlier fusion techniques (e.g.concatenation of both feature vectors, element-wise multiplication), the system could not fully interact with multimodal features. In the case of bilinear pooling, the system fully captures the complex association between image and textual features to get a more fine-grained classification decision. Each element of the textual feature interacts with every element of the visual feature using an outer product. The outer product of two vectors $T_i = (t_1, t_2, ...t_m)$ and $V_i = (v_1, v_2, ...v_n)$ can be defined as $\otimes$ which results in a matrix P $\in$R$^{(m\times n)}$. Here, T_i in $\mathbb{R}^m$ is the textual feature vector, and V_i in $\mathbb{R}^n$ is the visual feature vector.

$$M = T_i \otimes V_i = T_i \times V_i^T \tag{15}$$

where M_{m*n} is the output of the bilinear model.

3.7.2 Fusion of textual and visual features with Multimodal Factorized Bilinear(MFB)

Although bilinear pooling adequately captures element-wise interactions between feature dimensions, it does so at the expense of a set of parameters, which can result in significant computing costs and the risk of over-fitting. The Multimodal Factorized Bilinear(MFB) module may be used to fix this problem effectively. The association between textual and visual feature representations is maximised using this fusion mechanism. Given two feature vectors $T_i = (t_1, t_2, ...t_m)$ and $V_i = (v_1, v_2, ...v_n)$, where T_i is textual feature and V_i is visual feature of a meme. We can easily compute the bilinear pooling of these two vectors as follows:

$$M = T_i^T W_i V_i \tag{16}$$

where $W_i \in \mathrm{R}^{(m \times n)}$ is a projection matrix and M_i is the output of the bilinear model.

Furthermore, the projection matrix W_i in Eq.16 can be easily factorized into low-rank matrices.

$$\begin{aligned} M = T_i^T X_i Y_i^T V_i = \sum_{d=1}^{k} T_i^T x_d y_d^T V_i \\ = 1^T (X_i^T T_i \circ Y_i^T V_i) \end{aligned} \tag{17}$$

where k is the latent dimensionality of the factorized matrices X_i=$[x_1, x_2, ..., x_k]$ Y_i = $[y_1, y_2, ..., y_k]$, $\circ$ is the Hadamard product of two vectors, $1 \in \mathrm{R}^k$ an all-one vector. In order to obtain the output feature $M_{TV} \in \mathrm{R}^o$ by Eq.17, weights to be learned are two three-order tensors $X = [X_1, X_2, ..., X_o] \in \mathrm{R}^{(m \times k \times o)}$ and $Y = [Y_1, Y_2, ..., Y_o] \in \mathrm{R}^{(n \times k \times o)}$. We can easily reformulate X and Y vectors in $2-D$ matrices X'$\in \mathrm{R}^{m \times ko}$ and Y'$\in \mathrm{R}^{n \times ko}$ easily with simple reshape operation. We can then write Eq.17 as the following:

$$M_{TV} = \mathit{AvgPool}(X_i^T T_i \circ Y_i^T V_i, k) \tag{18}$$

$$M_{TV} = sign(M_{TV})|M_{TV}|^{0.5} \tag{19}$$

$$M_{TV} = (M_{TV})^T / ||M_{TV}|| \tag{20}$$

where AvgPool in Eq.18 is the average pooling over M_{TV}. Furthermore, to reduce the cost of variation in the magnitude of output neurons due to element-wise multiplication, Power-Normalization in Eq.19 and l_2-Normalization in Eq.20 is introduced to the MFB module. These operations restrict the model to go under the state of local minima.

We can formulate the class prediction for a given meme M_i using Softmax as the activation function in the final output layer as:

$$\hat{y} = P(Y_i | M_i, W, b) = \mathit{softmax}(M_i W_i + b_i) \tag{21}$$

where, $\hat{y}$ is the prediction probability of selecting the i_{th} class (Y_i) given M$_i$, bias b_i, and weight matrix W_i $(i \in$(neg,neu,pos)). We use the categorical cross entropy as loss function with the following formula:

$$\mathcal{L} = -\sum [y \log \hat{y} + (1 - y) \log(1 - \hat{y})] \tag{22}$$

where, y is the original class and $\hat{y}$ is the predicted class of the meme.

3.8 Models

For our experiment, we develop the following models. The first is the official baseline model from SemEval, whereas the others are different variations of our proposed system.

Model 1 (Baseline Model) This model is the baseline model reported in SemEval2020 Task8 Subtask-A paper(Sharma et al., 2020). This model uses CNN + BiLSTM framework to extract the textual features and VGG-16 to extract the visual elements.

Model 2 (Only Text) The first model is the baseline model for the text part of memes. We use the CNN architecture to obtain the textual features. The framework of this model is discussed in Section 3.2. This textual feature is passed through the output layer with one softmax activation for the final prediction.

Model 3 (Text+ self-Attention) In this model, we use self-attention on the top of text features. The framework of this model is described in Section 3.3. The attended textual feature is passed through one dense layer, following a softmax layer with the final prediction.

Model 4 (Only Image) The architecture of this model is given in Section 3.4. This model uses a pre-trained VGG19 framework to obtain the region-specific features without attention to classify a meme into a specific category. Extracted visual features are fed to the output layer having softmax activation for the final prediction.

Model 5 (Image+ self-Attention) In this model, attention is used on the top of the visual features as mentioned in Section 3.5. After applying attention to the visual feature, it is passed through a softmax layer with three output neurons.

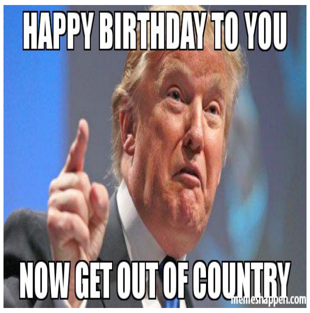
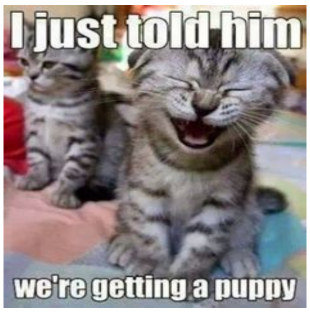
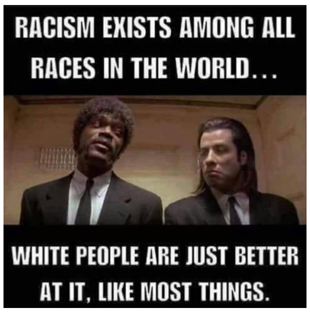
Input image			 RACISM EXISTS AMONG ALL RACES IN THE WORLD... WHITE PEOPLE ARE JUST BETTER AT IT, LIKE MOST THINGS.
	(a)	**(b)**	**(c)**
True label	negative	neutral	negative
Model2	positive	neutral	positive
Model3	neutral	neutral	neutral
Model4	negative	positive	positive
Model5	negative	negative	positive
Model6	neutral	positive	neutral
Model7	positive	positive	neutral
Model8	negative	neutral	negative

Figure 4: Outputs from different models

Model 6 (Text + Image with self-attention) In this model, we concatenate both textual T_i and visual feature V_i (c.f. Section 3.6). After obtaining a concatenated feature vector, we pass it through one dense layer. The dense layer follows the output layer with one softmax activation for the final prediction.

Model 7 (Proposed Model-1 (Text + Image with co-attention and bilinear pooling)) This is our first proposed model, described in Section 3.7.1.

Model 8 (Proposed Model-2 (Text + Image with co-attention and MFB)) This is our second proposed model which is described in Section 3.7.2.

Table 2: Data statistics

Data	Class	Statistics	Distribution
Train	Positive	4160	59.5%
	Neutral	2201	31.5%
	Negative	631	9.0%
	Total	6992	100%
Test	Positive	831	59.56%
	Neutral	439	31.44%
	Negative	126	9.02%
	Total	1396	100%

4 Datasets and Experiments

4.1 Datasets

To assess the significance of our proposed framework, we use a multimodal dataset given in SemEval2020 Task8 Sabtask A (Sharma et al., 2020). The dataset consists of 6,992 memes for training, where 10% is selected for validation, and testing is done on 1396 memes. Table 2 presents the summary of the dataset used.

4.2 Experimental Setup

For the experimental setup, we use keras with tensorflow at the backend. From the dataset distribution, it is visible that the dataset is skewed towards a positive class. To tackle this problem, we use the class weights for each class during implementation. We evaluate our system performance on the batch size of (16,32,64) and dropout rate as (0.2,0.3,0.4). We obtained the best performance using the batch size of 64 and the dropout rate of 0.4. During the training time for every model, we use the Adam optimizer with lr=3e-5, beta1=0.9, and beta2=0.999 for the loss optimization.

4.3 Result and Analysis

In this section, we discuss the performance of each model described in the above section. We report the results in the form of accuracy and F1-score. In Table 1, results of all the models are shown. The

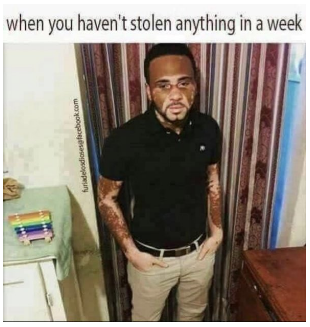
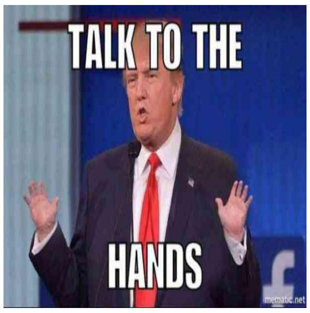
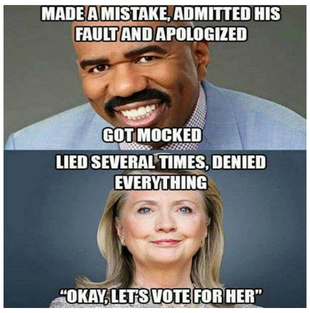
Input image			 MADE A MISTAKE, ADMITTED HIS FAULT AND APOLOGIZED GOT MOCKED LIED SEVERAL TIMES, DENIED EVERYTHING "OKAY, LET'S VOTE FOR HER"
True label	negative	negative	negative
Model 8	neutral	positive	neutral

Figure 5: Examples of miss-classification by the proposed framework

baseline model for text data obtained 49% accuracy with 32.78% F1-score. In contrast, when we applied attention to the top of the textual feature reported in Model 3, it gave us a decent baseline with 49% and 33.97% accuracy and F1-score, respectively. Similarly, Model 4 is a decent baseline model for only an image as an input with reported 29.36% accuracy and 34% F1-score. Furthermore, the visual attention model, i.e., Model 5, also demonstrates a comparatively good performance with a reported accuracy of 31.66%.

Table 3: Confusion matrix of the proposed model

	Negative	**Neutral**	**Positive**
Negative	26	18	82
Neutral	70	114	256
Positive	126	190	514

Model 6 is the framework where we merely concatenate the attended textual and visual feature vectors. We can observe in Table 1 that simple concatenation does not help the classifier to classify memes into its' right category effectively. The reported accuracy and F1-score for this model are 51% and 32.22%, respectively, which are −1.75% and +0.56% points increments (in terms of F1-score) when compared to Model 3 and Model 5, respectively.

To obtain a more robust multimodal classifier, we use our proposed deep learning framework mentioned in Section 3.7. Our model reported in Section3.7.1 i.e Model 7 performs better than all previously reported models. It shows +3.1% improvement in F1- score in comparison to Model 6. Furthermore, the significant growth in the accuracy as well as in the F1-score clearly shows the effectiveness of our proposed Model 8 mentioned in Section 3.7.2. We found the performance of Model 8 to have increased significantly in terms of F1-score by +4.59% and +1.49% when compared with Model 6 and Model 7, respectively. We find this improvement statistically significant as we performed the significance t-test conducted at a 5% significance level.

4.4 Detailed Analysis

We perform detailed quantitative and qualitative analysis of the output generated from our models to understand where our model succeeds and where our model fails. Results of all models are shown in Table 1. We take some example cases in Figure 4 where we evaluate the performance of all the models.

- For example (a), we can see that the visual model and our proposed model (text + visual) performs better than the other models.
- For example (b), it is shown that the textual model and our proposed model (text + visual) 3.7.2 performs well.
- In a similar way, for a given example (c), only the proposed model 3.7.2 shows the accurate output.

In Table 3, we report the confusion matrix of our proposed model. From the confusion matrix, we can identify the effectiveness of our proposed model. We can observe that using co-attention and an effective MFB based fused feature representation, the system can correctly capture the complex association between visual and textual representation.

We also perform qualitative analysis on the dataset to analyze the output from our proposed model. We observe that due to the implicit nature of negative polarity memes, in few cases, our proposed multimodal system couldn't relate the textual and visual features properly, which results in miss-classification. We encountered a few complex examples where both textual and visual parts were neutral separately, but it became negative when we combined both the modalities. In such cases, our model was not able to produce a good result (c.f. Figure 5).

5 Conclusion

In this work, we have proposed an end-to-end CNN-based deep neural network that consists of co-attention that jointly reasons about textual and visual representation. Additionally, we incorporated one common portrayal of a meme by utilizing the multimodal factorized bilinear pooling of textual and visual features. By introducing these joint representations, we obtain more effective multimodal features to identify the sentiment of a given meme. From the quantitative and qualitative error analysis on the recently released *SemEval-2020 Task-8* (Sharma et al., 2020) dataset, we observed that our proposed method produces promising results with respect to the baseline models. In the future, we will investigate more fusion strategies to combine both the modalities effectively; and investigate methods to extract essential objects from the meme.

References

Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca J. Passonneau. 2011. Sentiment analysis of twitter data.

Farman Ali, Daehan Kwak, Pervez Khan, Shaker El-Sappagh, Amjad Ali, Sana Ullah, Kye Hyun Kim, and Kyung-Sup Kwak. 2019. Transportation sentiment analysis using word embedding and ontology-based topic modeling. *Knowledge-Based Systems*, 174:27–42.

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. *Proceedings of the 26th International Conference on World Wide Web Companion*.

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.

Damian Borth, Rongrong Ji, Tao Chen, Thomas Breuel, and Shih-Fu Chang. 2013. Large-scale visual sentiment ontology and detectors using adjective noun pairs. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, page 223–232, New York, NY, USA. Association for Computing Machinery.

Alexandra Cernian, Valentin Sgarciu, and Bogdan Martin. 2015. Sentiment analysis from product reviews using sentiwordnet as lexical resource. In *2015 7th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pages WE–15–WE–18.

Thomas Davidson, Dana Warmsley, Michael W. Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *ICWSM*.

R. Dawkins. 2016. *The Selfish Gene*. Oxford Landmark Science. Oxford University Press.

Andrea Esuli and Fabrizio Sebastiani. 2006. SENTIWORDNET: A publicly available lexical resource for opinion mining. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

Xing Fang and Justin Zhan. 2015. Sentiment analysis using product review data. *J Big Data*, 2.

Paula Fortuna, João Rocha da Silva, Juan Soler-Company, Leo Wanner, and Sérgio Nunes. 2019. A hierarchically-labeled portuguese hate speech dataset.

Vasileios Hatzivassiloglou and Kathleen McKeown. 1997. Predicting the semantic orientation of adjectives. In *ACL*.

Anthony Hu and Seth Flaxman. 2018. Multimodal sentiment analysis to explore the structure of emotions. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*.

Sardar Haider Waseem Ilyas, Zainab Tariq Soomro, Ahmed Anwar, Hamza Shahzad, and Ussama Yaqub. 2020. Analyzing brexit's impact using sentiment analysis and topic modeling on twitter discussion. In *The 21st Annual International Conference on Digital Government Research*, dg.o '20, page 1–6, New York, NY, USA. Association for Computing Machinery.

Walter Kasper and Mihaela Vela. 2011. Sentiment analysis for hotel reviews.

Vishal Keswani, Sakshi Singh, Suryansh Agarwal, and Ashutosh Modi. 2020. Iitk at semeval-2020 task 8: Unimodal and bimodal sentiment analysis of internet memes.

Efthymios Kouloumpis, Theresa Wilson, and Johanna D. Moore. 2011. Twitter sentiment analysis: The good the bad and the omg! In *ICWSM*.

Akshi Kumar, Kathiravan Srinivasan, Wen-Huang Cheng, and Albert Zomaya. 2020. Hybrid context enriched deep learning model for fine-grained sentiment analysis in textual and visual semiotic modality social data. *Information Processing Management*, 57.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking aggression identification in social media. In *TRAC@COLING 2018*.

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, page 2267–2273. AAAI Press.

Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature Cell Biology*, 521(7553):436–444. Funding Information: Acknowledgements The authors would like to thank the Natural Sciences and Engineering Research Council of Canada, the Canadian Institute For Advanced Research (CIFAR), the National Science Foundation and Office of Naval Research for support. Y.L. and Y.B. are CIFAR fellows. Publisher Copyright: © 2015 Macmillan Publishers Limited. All rights reserved.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, page 375–384, New York, NY, USA. Association for Computing Machinery.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *NIPS*.

Navonil Majumder, Devamanyu Hazarika, Alexander F. Gelbukh, Erik Cambria, and Soujanya Poria. 2018. Multimodal sentiment analysis using hierarchical fusion with context modeling. *CoRR*, abs/1806.06228.

K. Mouthami, K. Nirmala Devi, and V. Murali Bhaskaran. 2013. Sentiment analysis and classification based on textual reviews. In *2013 International Conference on Information Communication and Embedded Systems (ICICES)*, pages 271–276.

Benet Oriol Sabat, Cristian Canton-Ferrer, and Xavier Giró i Nieto. 2019. Hate speech in pixels: Detection of offensive memes towards automatic moderation. *ArXiv*, abs/1910.02334.

Chhavi Sharma, Deepesh Bhageria, William Scott, Srinivas PYKL, Amitava Das, Tanmoy Chakraborty, Viswanath Pulabaigari, and Björn Gambäck. 2020. Semeval-2020 task 8: Memotion analysis - the visuo-lingual metaphor! *CoRR*, abs/2008.03781.

Han-Xiao Shi and Xiao-Jun Li. 2011. A sentiment analysis model for hotel reviews based on supervised learning. In *2011 International Conference on Machine Learning and Cybernetics*, volume 3, pages 950–954.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition.

Ha-Nguyen Tran and Erik Cambria. 2018. Ensemble application of elm and gpu for real-time multimodal sentiment analysis. *Memetic Computing*, 10.

Quoc-Tuan Truong and Hady W. Lauw. 2019. Vistanet: Visual aspect attention network for multimodal sentiment analysis. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):305–312.

Wei Wei and Jon Gulla. 2010. Sentiment learning on product reviews via sentiment ontology tree. pages 404–413.

Caiming Xiong, Victor Zhong, and Richard Socher. 2018. Dynamic coattention networks for question answering.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012. Learning from bullying traces in social media. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 656–666, Montréal, Canada. Association for Computational Linguistics.

Fan Yang, Xiaochang Peng, Gargi Ghosh, Reshef Shilon, Hao Ma, Eider Moore, and Goran Predovic. 2019. Exploring deep multimodal fusion of text and photo for hate speech classification. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 11–18, Florence, Italy. Association for Computational Linguistics.

Zhou Yu, Jun Yu, Yuhao Cui, Dacheng Tao, and Qi Tian. 2019. Deep modular co-attention networks for visual question answering. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6274–6283.

How Effective is Incongruity? Implications for Code-mix Sarcasm Detection

Aditya Shah [*]
CSE department
IIT Indore
India
adishah3103@gmail.com

Chandresh Kumar Maurya
CSE department
IIT Indore
India
chandresh@iiti.ac.in

Abstract

The presence of sarcasm in conversational systems and social media like chatbots, Facebook, Twitter, etc. poses several challenges for downstream NLP tasks. This is attributed to the fact that intended meaning of a sarcastic text is contrary to what is expressed. Further, the use of code-mix language to express sarcasm is increasing day by day. Current NLP techniques for code-mix data have limited success due to the use of different lexicon, syntax, and scarcity of labeled corpora. To solve the joint problem of code-mixing and sarcasm detection, we propose the idea of capturing incongruity through sub-word level embeddings learned via fastText. Empirical results show that our proposed model achieves F1-score on code-mix Hinglish dataset comparable to pretrained multilingual models while training **10x** faster and using *lower memory footprint*.

1 Introduction

Sarcasm is defined as a sharp remark whose intended meaning is different from what it looks like. For example, "*I am not insulting you. I am describing you.*" could mean that the speaker is insulting the audience, but the receiver does not get it. Sarcasm usually involves ambivalence (also known as *incongruity* which means words/phrases having contradictory implications (Xiong et al., 2019) and difficult to comprehend. Though English is used as a way to communicate and exchange messages, majority of the people still use the mother language to express themselves on social media (Danet et al., 2007). According to one study (Hong et al., 2011), more than 50% posts on Twitter are written in a language other than English. *Code-switching* (also known as *code-mixing*) is a writing style in which the author uses words from different languages either in the same sentence (called *intra–sentential*) or different sentences (called *inter–sentential*) switching. An example of code-switching is: *"he said kal karte hai kaam""* (Gloss: he said tomorrow we'll do the work). The studies such as (Vizcaíno, 2011; Siegel, 1995) show that people use code-switch language when trying to convey comicality, satire or humor. Motivated by previous studies, we plan to detect sarcasm in code-mix languages. Though many studies exist to detect sarcasm in unimodal and multimodal data (Joshi et al., 2015, 2017; Carvalho et al., 2009; Xiong et al., 2019; Cai et al., 2019), methods to detect sarcasm in code-mix data are limited and have not been explored much. (Bansal et al., 2020; Aggarwal et al., 2020; Swami et al., 2018). It is due to several challenges such as *ambiguous words, variable lexical representation, word-level code-mixing, reduplication*, and *word-order*.

To solve some of these issues, we present a deep-learning based architecture to capture incongruity in code-mix data. Our proposed model achieves competitive performance as compared to pre-trained multilingual models (fine-tuned on the code-mix sarcasm detection task) with significantly *fewer parameters and faster training time*. Our contributions are as follows:

- Propose a deep learning based architecture along with sub-word level features to capture incongruity for sarcasm detection.

- Evaluate the performance of the proposed model on the Hindi-English (Hinglish) code-mix Twitter data that we collected. We further analyze existing multilingual models on the same. Our code+data will be available on [1].

- We will release the benchmark sarcasm

[*]Work done while the author was an intern at IIT Indore

[1]https://github.com/likemycode/codemix

dataset for Hinglish language to facilitate further research on code-mix NLP.

2 Related Works

2.1 Learning Representation for Code-Mix Data

Models developed for multilingual representation learning have been explored for code-mix data representation by several authors (Winata et al., 2021; Khanuja et al., 2020; Aguilar et al., 2020a; Winata et al., 2018). Character-level representations have been utilized to address the out-of-vocabulary (OOV) issue in code-switch text (Winata et al., 2018), hand-crafted features were used in (Aguilar et al., 2019) for handling low-resource scenarios. Fine-tuning multilingual models like mBERT has shown to yield good results for various NLP tasks like Named-entity recognition (NER), part-of-speech (POS) tagging, etc., in (Khanuja et al., 2020), and surprisingly outperforms cross-lingual embeddings. Meta-embedding and hierarchical meta-embeddings have been found to be useful for closely-related language pairs in code-mix data (Winata et al., 2021) and usually outperform the mBERT (Khanuja et al., 2020). Char2Subword model proposed by (Aguilar et al., 2020b) builds representations from characters out of the subword vocabulary, and uses them to replace subwords in code-mix text (Winata et al., 2018), hand-crafted features were used in (Aguilar et al., 2019) for handling low-resource scenario. A centralized benchmark for Linguistic Code-switching Evaluation (LinCE) is released in (Aguilar et al., 2020a; Khanuja et al., 2020). Both of these works present results on several NLP tasks but *sarcasm detection*.

2.2 Sarcasm Detection in Code-Mix Data

There exists only a few works targeted towards sarcasm detection in code-mix data. In (Aggarwal et al., 2020), the author experiments with FastText (Joulin et al., 2016) and Word2Vec embeddings on two kinds of data: (1) Hinglish (Hindi-Eng) tweets, and (2) Hinglish+English tweets. They find that that Hinglish+English combination produces better results and achieves best F1 score of 79.4%. Various hand-crafted features, such as char n-grams, word n-grams etc., combined with random-forest/SVM are explored in (Swami et al., 2018) for sarcasm detection in code-mix, data and achieve F1-score of 78.4%. However, the dataset used is highly imbalanced with just 10% of sarcastic tweets and rest non-sarcastic. In such a scenario, the model might be biased towards predicting non-sarcastic tweets, and hence the evaluation results are quite skewed. Along similar lines, different switching features are used to form feature-vector and fed into a hierarchical attention network in (Bansal et al., 2020). They find that switching feature is a good indicator for irony/sarcasm/hate speech detection. However, none of these works handles *incongruity* explicitly or implicitly which has shown to achieve impressive results in sarcasm detection (Xiong et al., 2019).

3 Model Architecture

Code-mix language contains noisy words mixed with different languages and this might lead to out-of-vocabulary $< OOV >$ tokens. So, we use FastText skipgram (Bojanowski et al., 2016; Grave et al., 2018) for learning *subword level representation* from the code-mix data. We hypothesise that subword level representation is able to handle *ambiguous words*, *variable lexical representation*, and *word-level code-mixing*. For example, the ambiguous word "to" may be present in both Hindi and English language. Learning subword representation alleviates the problem of encoding the word "to" differently for both the languages. Further, variable length words like {"gharr", "gharrr", "gharrrr" will be split into tokens {"gha","har", "arr","rrr" } and uniquely represented using only these subwords tokens. Code-mix words like "chapless" (Mix of Bengali "chap" and English "less") are also represented via sub-words "chap" and "less". The proposed model architecture is shown in Fig. 1.

Each sentence s is represented by its embedding $E = [e_1^T, e_2^T, \ldots, e_n^T]$, where $e_i \in R^d$ is the embedding vector and n is the length of the sentence. Inspired by the work of (Xiong et al., 2019), we propose the use of self-matching network in order to capture *incongruity* within the code-mix sentence. Specifically, for word-embedding pairs (e_i, e_j), we first calculate the joint feature vector $m_{i,j}$ via

$$m_{i,j} = GELU(e_i^T \cdot M_{i,j} \cdot e_j) \qquad (1)$$

where $M_{i,j} \in R^{d \times d}$ is the weight parameter matrix (learnable) and *GELU* (Hendrycks and Gimpel, 2016) is Gaussian Error Linear Unit activation function. Instead of *tanh* as used in (Xiong et al., 2019), we use *GELU* as it provides well-defined gradients in the negative region. Compared to RELU, since GELU is differentiable for all input values so it is

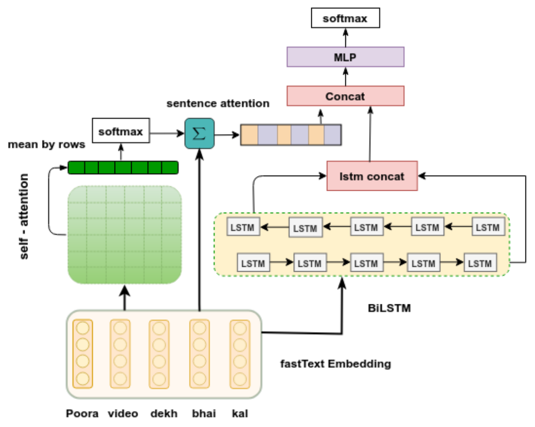


Figure 1: The model architecture

widely used in state-of-the-art NLP architectures. Our findings suggest that GELU activation yields better results for attention based mechanisms. Note that the above formulation is a form of *bilinear similarity* popularly used in *metric learning*. To calculate the attention score $\alpha_i, i \in (1, 2, \ldots, n)$ for each word, we take the mean of each row (contrary to *max* of rows as in (Xiong et al., 2019)) and apply the softmax for normalization.

$$\alpha_i = softmax(\mu(m_{1,i}), \mu(m_{2,i}), \ldots, \mu(m_{n,i})) \tag{2}$$

where $\mu()$ is the mean function, $m_{1,i}$ captures the incongruity of the first word with every other i^{th} word in the input text. Using the mean of all attention scores considers all the incongruous words present in the sentence for the computation. This helps the model to attend and learn from much larger span of incongruity. These attention scores denote how much weight should be assigned to each incongruous word. Next we calculate the weighted sentence attention vector v by:

$$v = \alpha^T E \tag{3}$$

Self-attention approach though captures the *incongruity* in the sentence, it misses the sentence's compositionality which is essential for sarcasm detection as suggested in (Tay et al., 2018). Therefore, sentence embedding is passed to the BiLSTM encoder (Graves et al., 2013) and the hidden states of the forward LSTM and backward LSTM are concatenated as

$$h_i = [\overrightarrow{LSTM}(e_i), \overleftarrow{LSTM}(e_i)], \forall i \in (1, 2, \ldots, n) \tag{4}$$

The output of the BiLSTM is concatenated with the sentence attention vector v and passed through the MLP layers with dropout. Finally, we pass it through softmax to predict the distribution over the binary labels (sarcasm vs non-sarcasm).

$$\hat{y} = softmax(MLP([v, h_n])) \tag{5}$$

where h_n is the hidden state corresponding to the last word in the forward and backward LSTM.

4 Experiments

4.1 The Dataset

The code-mix dataset used by (Aggarwal et al., 2020) is highly imbalanced with just 10% of sarcastic tweets and rest non-sarcastic. So, we create a dataset using TweetScraper built on top of scrapy [2] to extract code-mix hindi-english tweets. We pass search tags like #sarcasm, #humor, #bollywood, #cricket, etc., combined with most commonly used code-mix Hindi words as query. All the tweets with hashtags like #sarcasm, #sarcastic, #irony, #humor etc. are treated as positive. Non sarcastic tweets are extracted using general hashtags like #politics, #food, #movie, etc. The balanced dataset comprises of 166K tweets. We preprocess and clean the data by removing urls, hashtags, mentions, and punctuation in the data.

4.2 Baselines

The following baselines are used for comparison. (a) **Attention BiLSTM** (Aggarwal et al., 2020): The text features are extracted using word2vec and FastText which is fed to Series CNN, Parallel CNN, LSTM, Bi-LSTM and Attention Bi-LSTM, (b) **Multilingual Models**: To showcase the competitiveness of the proposed approach, we also compare with the state-of-the-art multilingual models like XLM-RoBERTa[3] and mBERT[4] from Huggingface library (Wolf et al., 2019). Specifically, we first fine-tune these models on the preprocessed code-mix corpus for mask language modeling task. Next, we use trained model by attaching a dense layer on top of it for detecting sarcasm in the code-mix tweets.

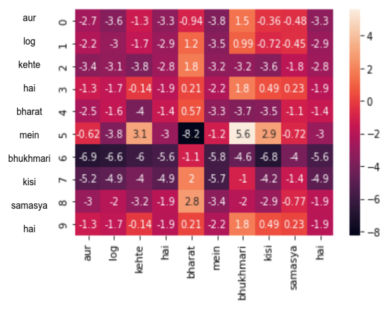
4.3 Experimental setup

For all the experiments, we use a train/valid/test split of 65:15:20. Categorical cross-entropy loss is minimized using adam optimizer for 15 epochs and learning rate of 5e-4 with step wise learning

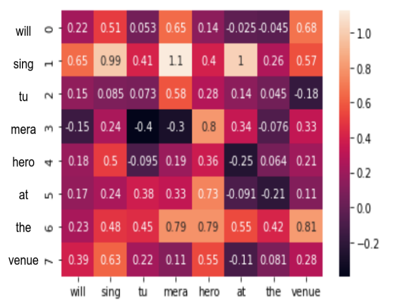
[2]https://github.com/jonbakerfish/TweetScraper
[3]https://tinyurl.com/ydseww9d
[4]https://tinyurl.com/2dafn48n

Table 1: Comparative evaluation of the proposed approach.

Model	Recall	Prec.	Acc.	F1	Params	GPU	Train time
Attn. BiLSTM	77.34	81.24	80.21	79.34	21M	68 MB	0.8Hr
XLM-RoBERTa	86.17	91.48	89.04	88.75	278M	575 MB	8 Hr
mBERT	83.20	**94.55**	**89.17**	88.51	167M	483 MB	7 Hr
SelfNet (Ours)	**88.12**	88.25	89.04	**88.89**	35M	80 MB	1 Hr



(a) Sarcastic



(b) Non Sarcastic

Figure 2: Incongruity visualization via attention matrix

rate scheduler. FastText embedding size is 100 and the number of hidden units in BiLSTM and MLP layers are 256. We apply dropout of 0.4 along with gradient clipping of 0.3.

4.4 Results

The comparative evaluation results are shown in Table 1. We can see that the proposed approach achieves better F1 score than the baselines on the Hinglish code-mix data. In particular, our approach achieves around 10 points more F1 score than the Attn. BiLSTM of (Aggarwal et al., 2020). Additionally, it achieves slightly better F1 score than the pre-trained multilingual models.

Further, we also provide a comparison among all the 4 models in terms of the trainable parameters, GPU memory and training time in Table 1. As it can be seen that multilingual models require much larger memory and use almost 10x more parameters than our approach. Also, it is worth noting that the dataset size used to train our model is significantly less than the dataset size used to train multilingual models. Based on the comparison, we observe that our proposed model achieves a good balance between performance and model size for code-mix sarcasm detection. Figure 2 illustrates the output raw attention matrix P obtained before applying activation to visualize incongruous words.

As we can see from the Figure 2a, the words *bharat, mein, and bhukhmari* hold the highest incongruity (negative values). These 3 words define the semantics of the sentence and our model correctly attends to those words while finding the incongruity. Similarly for Figure 2b, there's no such incongruity present in the text. Thus the model does not assign high negative scores to this matrix.

4.5 Ablation Study

To evaluate the effectiveness of the network, we conduct an ablation study on the proposed architecture. This is summarized in Table 2. We test the self matching network proposed by (Xiong et al., 2019) which is referred to as Self Matching Net. Next, we replace biLSTM in our model with XLM-RoBERTa and mBert. The resulting models are denoted by "with XLM-RoBERTa" and "with mBert" respectively. The original Self Matching network does not perform so well on code-mix data as it only considers the most incongruous word pairs for prediction. Using mean operation helps to capture all the incongruous words which results in performance gain. Next when we replace BiLSTM with the multilingual models, the resulting approaches do not perform better than the proposed model. Although these models are trained on huge multilingual corpus, our study suggests that we can capture nuances of code-mix language using self-attention and simpler models like BiLSTM in a better way.

Table 2: Ablation Study

Models	Recall	Prec.	Acc.	F1
Self Matching Net	81.68	81.55	81.7	81.68
with XLM-RobertA	87.71	87.73	87.81	87.81
with mBERT	86.94	86.72	86.85	86.94
SelfNet (Ours)	**88.12**	**88.25**	**89.04**	**88.89**

5 Conclusion & Future work

In the present work, we propose the significance of incongruity in order to capture sarcasm in code-mix data. Our model effectively captures incongruity through FastText sub-word embeddings to detect sarcasm in the text. Empirical results on code-mix sarcasm data show that our approach performs satisfactorily compared to the multilingual models while saving memory footprint and training time. In future, we plan to work on a generalized model for other code-mix NLP tasks (NLI, NER, POS, QA etc) as well as test other code-mix languages like English - Spanish, English - Tamil, English - French etc.

References

Akshita Aggarwal, Anshul Wadhawan, Anshima Chaudhary, and Kavita Maurya. 2020. " did you really mean what you said?": Sarcasm detection in hindi-english code-mixed data using bilingual word embeddings. *arXiv preprint arXiv:2010.00310*.

Gustavo Aguilar, Sudipta Kar, and Thamar Solorio. 2020a. Lince: A centralized benchmark for linguistic code-switching evaluation. *arXiv preprint arXiv:2005.04322*.

Gustavo Aguilar, Suraj Maharjan, Adrian Pastor López-Monroy, and Thamar Solorio. 2019. A multi-task approach for named entity recognition in social media data. *arXiv preprint arXiv:1906.04135*.

Gustavo Aguilar, Bryan McCann, Tong Niu, Nazneen Rajani, Nitish Keskar, and Thamar Solorio. 2020b. Char2subword: Extending the subword embedding space from pre-trained models using robust character compositionality. *arXiv preprint arXiv:2010.12730*.

Srijan Bansal, Vishal Garimella, Ayush Suhane, Jasabanta Patro, and Animesh Mukherjee. 2020. Code-switching patterns can be an effective route to improve performance of downstream nlp applications: A case study of humour, sarcasm and hate speech detection. *arXiv preprint arXiv:2005.02295*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Yitao Cai, Huiyu Cai, and Xiaojun Wan. 2019. Multi-modal sarcasm detection in twitter with hierarchical fusion model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2506–2515.

Paula Carvalho, Luís Sarmento, Mário J Silva, and Eugénio De Oliveira. 2009. Clues for detecting irony in user-generated contents: oh...!! it's" so easy";-. In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 53–56.

Brenda Danet, Susan C Herring, Susan C Herring, et al. 2007. *The multilingual Internet: Language, culture, and communication online*. Oxford University Press on Demand.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. *ArXiv*, abs/1802.06893.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415.

Lichan Hong, Gregorio Convertino, and Ed Chi. 2011. Language matters in twitter: A large scale study. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 5.

Aditya Joshi, Pushpak Bhattacharyya, and Mark J Carman. 2017. Automatic sarcasm detection: A survey. *ACM Computing Surveys (CSUR)*, 50(5):1–22.

Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015. Harnessing context incongruity for sarcasm detection. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 757–762.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Simran Khanuja, Sandipan Dandapat, Anirudh Srinivasan, Sunayana Sitaram, and Monojit Choudhury. 2020. Gluecos: An evaluation benchmark for code-switched nlp. *arXiv preprint arXiv:2004.12376*.

Jeff Siegel. 1995. How to get a laugh in fijian: Code-switching and humor. *Language in Society*, pages 95–110.

Sahil Swami, Ankush Khandelwal, Vinay Singh, Syed Sarfaraz Akhtar, and Manish Shrivastava. 2018. A corpus of english-hindi code-mixed tweets for sarcasm detection. *arXiv preprint arXiv:1805.11869*.

Yi Tay, Luu Anh Tuan, Siu Cheung Hui, and Jian Su. 2018. Reasoning with sarcasm by reading in-between. *arXiv preprint arXiv:1805.02856*.

María José García Vizcaíno. 2011. Humor in code-mixed airline advertising. *Pragmatics*, 21(1):145–170.

Genta Indra Winata, Samuel Cahyawijaya, Zihan Liu, Zhaojiang Lin, Andrea Madotto, and Pascale Fung. 2021. Are multilingual models effective in code-switching? *arXiv preprint arXiv:2103.13309*.

Genta Indra Winata, Chien-Sheng Wu, Andrea Madotto, and Pascale Fung. 2018. Bilingual character representation for efficiently addressing out-of-vocabulary words in code-switching named entity recognition. *arXiv preprint arXiv:1805.12061*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Tao Xiong, Peiran Zhang, Hongbo Zhu, and Yihui Yang. 2019. Sarcasm detection with self-matching networks and low-rank bilinear pooling. In *The World Wide Web Conference*, pages 2115–2124.

Contrastive Learning of Sentence Representations

Hefei Qiu Wei Ding Ping Chen
University of Massachusetts Boston, Boston, MA
{hefei.qiu001, wei.ding, ping.chen}@umb.edu

Abstract

Learning sentence representations which capture rich semantic meanings has been crucial for many NLP tasks. Pre-trained language models such as BERT have achieved great success in NLP, but sentence embeddings extracted directly from these models do not perform well without fine-tuning. We propose Contrastive Learning of Sentence Representations (CLSR), a novel approach which applies contrastive learning to learn universal sentence representations on top of pre-trained language models. CLSR utilizes semantic similarity of two sentences to construct positive instance for contrastive learning. Semantic information that has been captured by the pre-trained models is kept by getting sentence embeddings from these models with proper pooling strategy. An encoder followed by a linear projection takes these embeddings as inputs and is trained under a contrastive objective. To evaluate the performance of CLSR, we run experiments on a range of pre-trained language models and their variants on a series of Semantic Contextual Similarity tasks. Results show that CLSR gains significant performance improvements over existing SOTA language models.

1 Introduction

Learning sentence representations that can encode semantic information is crucial for many Natural Language Processing (NLP) tasks such as question answering, summarization, machine translation. Many attempts have been made to learn general purpose sentence embeddings (Le and Mikolov, 2014; Kiros et al., 2015; Hill et al., 2016; Conneau et al., 2017; Arora et al., 2017; Logeswaran and Lee, 2018; Cer et al., 2018; Subramanian et al., 2018; Pagliardini et al., 2018). Since transformer-based pre-trained language models such as BERT are introduced and achieve the state-of-the-art results in many NLP tasks, several methods have been proposed to generate sentence embeddings with some pooling strategy such as mean, max from word level embeddings and fine tune these models on downstream tasks (Reimers and Gurevych, 2019) or train to calibrate these models for isotropic embeddings (Li et al., 2020; Su et al., 2021).

Inspired by the recent development of contrastive learning in learning visual representations (Chen et al., 2020), we design a contrastive learning based architecture to learn high-quality semantic sentence representations and show this approach can significantly improve the sentence representations. We integrate both pre-trained language models and contrastive learning and call our architecture Contrastive Learning of Sentence Representations (CLSR). Different from previous methods of using data augmentation to construct positive pairs in contrastive learning (Chen et al., 2020), we use semantic similarity or entailment relation of two sentences to build positive pairs. By sending two similar sentences into a pre-trained language model and then generate vector representations by pooling, CLSR is able to keep the rich information captured by these pre-trained models. A contrastive objective is used to train an encoder followed by a linear projection to further learn the embeddings in an unsupervised way. CLSR is model-agnostic. The initial sentence embeddings it takes as inputs can come from any pre-trained model.

2 Related Work

Learning sentence embeddings has attracted a lot of interest. Paragraph Vector (Le and Mikolov, 2014) and Skip-Thought (Kiros et al., 2015) learns generic, distributed sentence representations in an unsupervised fashion, one by proposing two log-bilinear models and the other one by training an encoder-decoder model to reconstruct the surrounding sentences of an encoded passage. Hill et al. (2016) proposed Sequential Denoising Autoencoders and FastSent to learn sentence representations from unlabelled data. InferSent (Conneau et al., 2017) performs the learning in a supervised way by training a Siamese BiLSTM network on Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015). Universal Sentence Encoder (Cer et al., 2018), with two variants, combines both by training with transfer learning on unsupervised data and being augmented on supervised data from the SNLI corpus (Bowman et al., 2015).

Recently there have been several attempts to improve sentence embeddings from pooled outputs of pre-trained language model such as BERT (Devlin et al., 2019). Sentence-BERT (Reimers and Gurevych, 2019) trains a Siamese network to fine-tune BERT and its variants. BERT-flow (Li et al., 2020) learns more smooth and isotropic embeddings by applying normalizing flow (Kumar et al., 2020) to convert BERT sentence embedding distribution into a Gaussian distribution. SBERT-WK (Wang and Kuo, 2020) improves Sentence-BERT by incorporating the pattern of layer-wised word representations in subspace. Su et al. (2021) applied whitening technique to enhance the isotropy of sentence representations.

Since contrastive learning has achieved great success in unsupervised representation learning in computer vision, it also has gained much interest in NLP including sentence representation learning. The following are some recent or concurrent work, many of which have explored different data augmentation strategies. IS-BERT (Zhang et al., 2020) learns through maximizing the mutual information between the global sentence representation and its local token representation. CERT (Fang and Xie, 2020) augments sentences using back-translation (Edunov et al., 2018). DeCLUTR (Giorgi et al., 2020) applies a contrastive objective on textual segments sampled from nearby in the same document. CLEAR (Wu et al., 2020) uses multi sentence-level augmentation to construct positive pairs. Carlsson et al. (2021) proposes Contrast Tension which counters the task biases in pre-trained language models by contrasting the noise between the output from two independent models. ConSERT (Yan et al., 2021) explores several ways to augment data such as adversarial attack, token shuffling etc. SimCSE (Gao et al., 2021) constructs positive instances by taking different outputs of the same sentence from the same pre-trained language model using dropout. Besides treating the task as unsupervised learning, although some of the above work such as Yan et al. (2021), Gao et al. (2021) also explore it as supervised learning, our method is mainly to show, by simply using sentence pairs with high similarity or entailment relation in existing labeled corpus to construct positive instances, contrastive learning can still further significantly improve the quality of sentence embeddings on top of any pre-trained language model.

3 Model

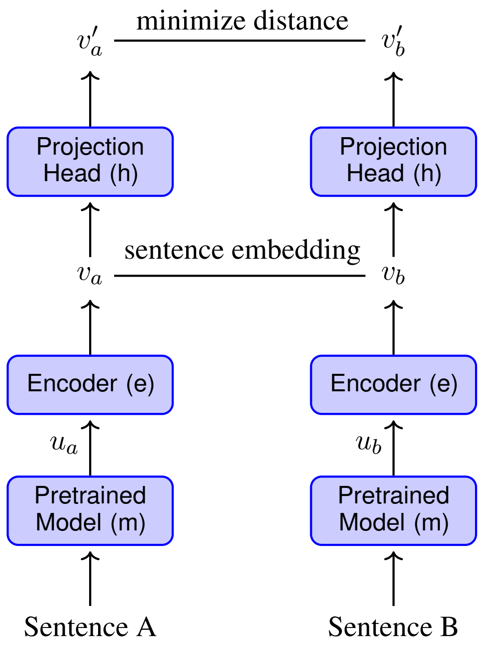


Figure 1: Contrastive learning of sentence representations. Two semantically similar sentences A and B (a positive pair) are sent to the same pre-trained model m (e.g., BERT). Each sentence embedding from m goes to an encoder e and a projection function h. A contrastive loss is applied to minimize the distance of two sentence embeddings. Output from e is used as the sentence representation for downstream tasks.

Contrastive learning has been a promising approach in self-supervised learning. It learns generic representations by contrasting positive pairs against negative pairs. SimCLR (Chen et al., 2020) is a simple framework for contrastive self-supervised learning of visual representations without using specialized architectures or a memory bank.

CLSR adopts contrastive learning framework SimCLR (Chen et al., 2020) to learn sentence semantic representations as shown in Figure 1. CLSR consists of:

- A pre-trained language model or pre-trained sentence encoder takes a raw sentence as input and output the sentence embedding, pooling may be applied. For example, pre-trained BERT with average pooling could be used to generate a vector representation for a sentence. Two sentences with high similarity are sent into the pre-trained model respectively to generate two sentence embeddings. They are considered as a positive pair. One is treated as a positive instance of the other. This step is different from SimCLR. We do not apply data augmentation technique to construct positive instance due to natural languages being highly discrete semantically. Instead, we use the property of semantic similarity.

- A neural network based encoder e further encodes sentence pairs into vectors respectively. This encoder could be of any structure. Since a well-pre-trained language model has been applied in the first step, in our setting, a simple Multi-Layer Perceptron (MLP) with 1 hidden layer and ReLu nonlinearity on the output is used to further encode information learned through contrastive learning without fine-tuning. The representation from this encoder is used for downstream tasks.

- A linear projection head h is applied to map sentence embeddings to a new representation space by training with contrastive objective. We follow the design in (Chen et al., 2020) which shows that a projection head can improve performance on downstream tasks.

- A contrastive loss function is defined as following: for a given set of sentences $\{s_m\}$ that contain positive sentence pair s_a and s_b with high semantic similarity, the contrastive learning process is to find s_b in $\{s_m\}_{b \neq a}$.

After randomly sampling m sentence pairs into a mini batch, this batch contains 2m sentences. Each sentence pair is treated as a positive pair, and the rest of sentences in the batch are treated as in-batch negatives (Chen et al., 2020; Henderson et al., 2017). We hypothesize that the probability of having one or more sentences in the negative samples that are highly semantically similar with that in the positive pair is very low and thus is ignored. Then the loss function of a positive sentence pair (s_a, s_b) is defined as:

Base Model		CLSR-STSB	CLSR-NLI
`BERT-base`	59.32	64.94	**76.74**
`BERT-large`	57.77	63.84	**78.75**
`SBERT-base`	77.12	80.15	**81.93**
`SBERT-large`	79.19	80.19	**83.76**

Table 1: Spearman correlations on STS-B development set. The 1^{st} column includes 4 base models and their performance on STS-B. The 2^{nd} and 3^{rd} columns include the performance of CLSR built on each base model and trained on STS-B or NLI data respectively.

Batch Size	64	128	256	512	1024
STS-B	64.93	67.38	69.78	76.74	78.55

Table 2: Spearman correlations on STS-Benchmark development set to show the effect of batch size. CLSR is built on BERT-base-uncased and trained on STS-B.

$$\ell_{s_a,s_b} = -log\frac{e^{sim(v_a,v_b)/\tau}}{\sum_{i=1}^{2m} I_{(a,i)} e^{sim(v_a,v_i)/\tau}}, \qquad (1)$$

where τ is the temperature hyper-parameter, sim(v_a, v_b) is a function measuring similarity between two given sentence vectors. We use cosine similarity for measurement. I is an indicator function to determine if a sentence s_i is included as a negative sample or not. If i $\neq$ a, it returns 1; otherwise, 0. The final loss is computed across all the positive sentence pairs in a mini batch. Positive pairs (s_a, s_b) and (s_b, s_a) are both included.

4 Experiments

4.1 Experiment Settings

Datasets: We use two types of training data. One is Semantic Textual Similarity (STS) in which STS-Benchmark is chosen. It comprises 8,628 sentence pairs with similarity score 0-5. We pre-process the data by keeping sentence pairs with scores higher than 4 which indicate high similarity. This gives us totally 1,406 pairs. The other one is Natural Language Inference (NLI) data. We follow Sentence-BERT (Reimers and Gurevych, 2019) to concatenate two NLI datasets: SNLI (Bowman et al., 2015) and Multi-Genre NLI (Williams et al., 2018). SNLI dataset contains 570k English sentence pairs labeled with entailment, contradiction, and neutral. While Multi-Genre NLI is a collection of 433k sentence pairs annotated with the same three labels.We pre-process them by only selecting sentence pairs with label entailment. This leads to total 314,315 samples used in our training.

Model	STS12	STS13	STS14	STS15	STS16	STS-B	Avg.
BERT-base	45.61	56.57	57.15	62.94	64.74	64.48	58.58
CLSR-BERT-base	**59.83**	**66.16**	**63.80**	**70.11**	**69.71**	**70.03**	**66.61**
BERT-large	46.98	52.88	49.56	56.63	61.64	65.37	57.51
CLSR-BERT-large	**60.02**	**63.19**	**62.74**	**68.81**	**71.78**	**73.53**	**66.68**
SBERT-base	66.35	73.76	**73.88**	77.33	73.62	**73.63**	73.10
CLSR-SBERT-base	**66.38**	**74.96**	73.81	**77.93**	**75.22**	70.35	**73.11**
SBERT-large	68.79	75.71	**75.12**	**80.29**	75.91	75.35	75.20
CLSR-SBERT-large	**69.49**	**76.74**	74.64	78.90	**77.84**	**76.84**	**75.74**

Table 3: Comparison of CLSR models and their corresponding base pre-trained models on the series of STS tasks. Spearman correlations multiplied by 100 are reported. SBERT-base and SBERT-large refer to Sentence BERT built on BERT base or large and trained using NLI datasets with mean pooling strategy.

4 SOTA models for comparison: For an accurate assessment, BERT-base, BERT-large, SBERT-base, SBERT-large are selected to compare with CLSR. BERT models are selected due to their good performance and popularity in NLP. SBERT models are popular pre-trained sentence embedding models and represent the best SOTA performance.

6 STS tasks for evaluation: Since CLSR can take sentence embeddings from any base models such as BERT as input, we evaluate its effectiveness by comparing CLSR and its base models on the same task. For example, when we perform evaluation on STS-B, if CLSR is trained by taking embeddings from pre-trained BERT base model with mean pooling over its last layer, we run both BERT and CLSR models on STS-B. As Pearson correlation is shown to be not suitable for STS (Reimers et al., 2016), we report the results of Spearman correlation between the cosine similarity of a sentence pair and the ground truth label. We evaluate our model on 6 Semantic Textural Similarity tasks that include STS12-STS16 (Agirre et al., 2012, 2013, 2014, 2015; Artetxe et al., 2016), the STS-Benchmark (Cer et al., 2017).

Model setting and hyper-parameters: To fully assess the computation efficiency of our approach, we use a simple MLP with only 1 hidden layer and a 768-dimensional output layer as the encoder network, and a linear projection head to project the presentation to a latent space but with the same dimensions. We train at batch size 512 for 2000 epochs. Learning rate is 0.5 with the decay rate of 0.0001. Temperature is 0.1. We adopt linear warmup in the first 10 epochs and decay learning rate with cosine decay schedule without restarts (Chen et al., 2020; Loshchilov and Hutter, 2016).

4.2 Training Set Construction

To construct positive instances for contrastive learning, sentence pairs with high similarity in STS tasks can be naturally used. Sentences with entailment relation in NLI tasks can be an alternative option. In order to decide which training dataset performs better, we run an experiment on selection of training data. CLSR are trained with the 4 base models on STS-B and NLI datasets respectively. Following convention, Spearman correlations are reported on the STS-B development set. As shown in Table 1, compared with base pre-trained models, CLSR built on those models achieve significant improvements overall. Models trained on NLI dataset perform much better than those trained on STS-B with 15 points increment over BERT-base. Based on this result, we will report results only on NLI dataset due to space limitation.

Pooling Strategy	STS-B
CLS	61.34
mean	76.74

Table 4: Comparison on STS Benchmark development set to show the effect of different pooling strategy. The CLSR is built on BERT-base-uncased.

4.3 Results on STS Tasks

We evaluate CLSR framework on a series of STS tasks. We run a CLSR model and its base pre-trained model on the tasks respectively. Spearman correlations are reported in Table 3.

Compared with pre-trained BERT base and large models, the corresponding CLSR models increase the performance on all STS tasks by large margins. Compared with SOTA SBERT models, CLSR also shows solid improvement. It's reasonable to infer that such an improvement could be more significant with more training data, as we only train the CLSR model using roughly 1/3 of the NLI data, while SBERT fine-tunes the BERT models using all the data from the same dataset. The pre-trained BERT models are not fine-tuned and only a simple 2-layer MLP is designed to further encode the sentence embedding. Surprisingly this simple approach can still slightly improve performance on several tasks compared with SBERT. This further validates the effectiveness of contrastive learning approach.

4.4 Ablation Study

Effect of batch size. Effect of batch size is shown in Table 2. The performance on STS-B development set shows that larger batch size brings better performance. This is consistent with the previous finding that contrastive learning benefits from large batch size (Chen et al., 2020). Since there are 2(n-1) negative instances in a mini-batch with size n, the change of batch size affects the number of negative instances more than that of positive instances. Thus it can be further inferred that, contrastive learning in the proposed framework learns more from larger number of negative instances.

Effect of pooling strategy. Previous work has shown the effect of pooling strategy (Xiao, 2018; Reimers and Gurevych, 2019). More specifically, taking the average of all the output word embeddings outperforms usage of the CLS token embedding as sentence embedding. This is also confirmed in our model as shown in table 4. By taking the mean of all the work embeddings from the last layer in BERT-base as input for CLSR, its performance on STS-Benchmark task increases as much as 15 points over the CLS token embedding.

5 Conclusion and Future Work

This paper presents a novel approach of applying contrastive learning on pre-trained language models to learn generic sentence representations. The evaluation on a series of STS tasks shows that our approach outperforms the pre-trained SOTA language models significantly. How to construct multiple positive instances and further integrate the idea of contrastive learning will be explored in future.

Acknowledgments

This work was supported by a NSF award IIS-1914489.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263, Denver, Colorado. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 81–91, Dublin, Ireland. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 task 6: A pilot on semantic textual similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393, Montréal, Canada. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA. Association for Computational Linguistics.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294, Austin, Texas. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. *CoRR*, abs/1508.05326.

Fredrik Carlsson, Amaru Cuba Gyllensten, Evangelia Gogoulou, Erik Ylipää Hellqvist, and Magnus Sahlgren. 2021. Semantic re-tuning with contrastive tension. In *International Conference on Learning Representations*.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic*

Evaluation (SemEval-2017), pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium. Association for Computational Linguistics.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. 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.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. *CoRR*, abs/1808.09381.

Hongchao Fang and Pengtao Xie. 2020. CERT: contrastive self-supervised learning for language understanding. *CoRR*, abs/2005.12766.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings.

John M. Giorgi, Osvald Nitski, Gary D. Bader, and Bo Wang. 2020. Declutr: Deep contrastive learning for unsupervised textual representations. *CoRR*, abs/2006.03659.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun hsuan Sung, Laszlo Lukacs, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367–1377, San Diego, California. Association for Computational Linguistics.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*, pages 3294–3302.

Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. 2020. Videoflow: A conditional flow-based model for stochastic video generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1188–1196, Bejing, China. PMLR.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the sentence embeddings from pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9119–9130, Online. Association for Computational Linguistics.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *International Conference on Learning Representations*.

Ilya Loshchilov and Frank Hutter. 2016. SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. 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 528–540, New Orleans, Louisiana. Association for Computational Linguistics.

Nils Reimers, Philip Beyer, and Iryna Gurevych. 2016. Task-oriented intrinsic evaluation of semantic textual similarity. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 87–96, Osaka, Japan. The COLING 2016 Organizing Committee.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Jianlin Su, Jiarun Cao, Weijie Liu, and Yangyiwen Ou. 2021. Whitening sentence representations for better semantics and faster retrieval. *CoRR*, abs/2103.15316.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning.

Bin Wang and C.-C. Jay Kuo. 2020. SBERT-WK: A sentence embedding method by dissecting bert-based word models. *CoRR*, abs/2002.06652.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. 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 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Zhuofeng Wu, Sinong Wang, Jiatao Gu, Madian Khabsa, Fei Sun, and Hao Ma. 2020. Clear: Contrastive learning for sentence representation.

Han Xiao. 2018. bert-as-service. `https://github.com/hanxiao/bert-as-service`.

Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. 2021. ConSERT: A contrastive framework for self-supervised sentence representation transfer. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5065–5075, Online. Association for Computational Linguistics.

Yan Zhang, Ruidan He, Zuozhu Liu, Kwan Hui Lim, and Lidong Bing. 2020. An unsupervised sentence embedding method by mutual information maximization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1601–1610, Online. Association for Computational Linguistics.

Classifying Verses of the Quran using Doc2vec

Menwa Alshammeri[1,2] **Eric Atwell**[2] **Mohammad Ammar Alsalka**[2]
[1]Jouf University, College of Computer and Information Sciences, Sakaka, Saudi Arabia
[2]University of Leeds, School of Computing, Leeds, UK
{scmhka, E.S.Atwell, M.A.Alsalka}@leeds.ac.uk

Abstract

The Quran, as a significant religious text, bears important spiritual and linguistic values. Understanding the text and inferring the underlying meanings entails semantic similarity analysis. We classified the verses of the Quran into 15 pre-defined categories or concepts, based on the Qurany corpus, using Doc2Vec and Logistic Regression. Our classifier scored 70% accuracy, and 60% F1-score using the distributed bag-of-words architecture. We then measured how similar the documents within the same category are to each other semantically and use this information to evaluate our model. We calculated the mean difference and average similarity values for each category to indicate how well our model describes that category.

1 Introduction

The richness of the Quran and the deep layers of its meaning offer immense potential for further study and experiments. The knowledge in the Quran was presented using different approaches, mainly using the tree-structure hierarchy (Ta'a et al., 2014). As a result, determining a concept's true meaning in the Quran is difficult. We want to classify the Quran verses based on topics or meanings to assist users in identifying the religious knowledge explained in the Quran. There has been previous work on classifying textual documents and sentences in English and Arabic (Al-Kabi et al., 2013). However, only a few studies in the literature attempt to classify the verses of the Holy Quran (Al-Kabi et al., 2013; Al-Kabi et al., 2005; Ta'a et al., 2014; Akour et al., 2014). Therefore, using NLP combined with ML, this paper presents an approach to classifying the Quran based on topics and meanings.

To do so, we need to compute the similarity in meaning between its passages. We focus on sentence/ paragraph levels. Therefore, we represent the verses of the Quran as vectors of features and compare them by measuring the distance between these features. We use Doc2ve[1] to compute features that capture the semantics of the Quranic verses. We then train a logistic regression classifier in a supervised way to learn the underlying meanings and classify the verses of the Quran into fifteen predefined classes or categories. We then use the cosine similarity measure on the vectors to examine how semantically similar the verses are in each class. We compute two metrics: average similarity and mean similarity difference to inspect the relation between the verses in the same class and other classes. This information indicates how more similar same-category documents are to each other than to documents from different categories. A higher average similarity indicates how similar the documents are in each category. A higher mean difference implies that the model can identify those documents in one class are more distinct from those in other classes. Since we are interested in a topical classification, we use the Qurany corpus to train and evaluate our model.

The rest of the paper is organized as follows: Section 2 presents studies related to the classification of the verses of the Quran. Section 3 describes our approach to classifying the Quranic verses and our experiments. Section 4 presents our evaluation and results. Finally, section 5 states our conclusions and future research directions.

[1]https://radimrehurek.com/gensim/models/doc2vec.html

2 Related Work

This section briefly reviews previous work conducted on the topical classification of holy Quran verses.

Hamed and Ab Aziz (2018) proposed a Quran classification using the Neural Network classifier based on the predefined topics. The study used the English translation of the Holy Quran. They applied the classification to Al-Baqarah chapter as it contains many commands and topics. They classified the verses of Al-Baqara into two classes, Fasting, and Pilgrimage. The thesis of Al-Kabi et al. (2013) is restricted in topical classification of only two Quran chapters: Fatiha (7 verses) and Yaseen (83 verses). Another study (Al-Kabi et al., 2005), evaluated the effectiveness of four well-known classification algorithms: Decision Tree, K-Nearest Neighbor (K-NN), Support Vector Machine (SVM) and Naïve Bayes (NB), to classify Quran verses according to their topics. They used the manual topical classification of Quranic verses by (Abu Al-Khair and Kabbani , 2003) to train and evaluate the four classifiers. Three selected topics (classes) are used, and 1,227 verses were used in this study out of 6236 verses in the whole Quran. Another classification has been presented by Qurany (Abbas, 2009). This project annotates the verses of the Qur'an with a comprehensive index of nearly1100 topics; it classifies the Qur'an into fifteen main themes and subdivides the main themes into sub-themes.

In this work, we exploit the distributed representation of text to capture the semantic properties of the 6236 Arabic verses of the Quran. We transformed the verses of the Quran into a numerical form, which can be used as input to ML methods to examine the semantic similarity between the Quranic verses and classify them into topical classes.

3 Methodology

The objective of this experiment is to evaluate our model in capturing the semantic properties of verses of the Quran. Therefore, we examine our model on the following tasks:

1. Classify the verses of the Quran into 15 pre-defined categories or classes using Doc2Vec and Logistic Regression.

2. Measure how similar the verses within the same category are to each other semantically, and use this information to evaluate our model.

3.1 The Data

For the purpose of training and testing our model, we create a dataset that contains the 6236 verses of the Quran categorized into 15 main topical themes; based on Qurany[2] corpus. Table 1 shows the high – level concepts from Qurany corpus.

Main Concept	English Concept
أركان الإسلام	Pillars of Islam
الإيمان	Faith
القصص والتاريخ	The Stories and The History
القرآن الكريم	The Holy Quran
العمل	The Work
الإنسان والعلاقات الأخلاقية	Man, and The Moral Relations
الإنسان والعلاقات الاجتماعية	Man, and The Social Relations
الجهاد	Jihad
العلوم والفنون	Science and Art
الديانات	Religions
تنظيم العلاقات المالية	Organizing Financial Relationships
الدعوة إلى الله	The Call for Allah
العلاقات القضائية	Judicial Relationships
التجارة والزراعة والصناعة والصيد	Trade, Agriculture, Industry and Hunting
العلاقات السياسية والعامة	General and Political Relationships

Table 1: The high-level concepts from the Qurany corpus

Each verse is annotated with a sequence of concepts besides the main concept/theme. The verses are split into training and test sets. Table 2 shows an example of the dataset.

[2] http://quranytopics.appspot.com

Chapter	Ayah	Verse Text	
2	3	الذين يؤمنون بالغيب ويقيمون الصلاة ومما رزقناهم ينفقون	
Translation		**Class**	**Trans**
Who believe in the Unseen, and establish worship, and spend of that We have bestowed upon them;		أركان الإسلام	Pillars of Islam

Table 2: Example of the verse annotation from the dataset

3.2 Classifying the Quran Verses using Logistic Regression

Following our approach in (Alshammeri et al., 2020), we map the verses of the Quran to numerical vectors. We use ML model on our vectorized verses to classify the Quran verses into the associated concepts or classes. We set up the train/ test documents, pre-process them to be ready for training and classification. We train the Doc2vec model using the training set and generate the vectors. We then build the vector representations for the classifier; we infer vectors for the documents in the test set using the trained model. Then we train the logistic regression classifier.

3.3 Testing Category-Wise & Cross-Category Verses Similarity

Documents belonging to the same category would seem to be more similar than documents belonging to different categories, intuitively. And that's how we judge our model: a good model should generate higher similarity values for verses in the same category than for verses from different categories.

4 Evaluation and Results

4.1 Classification Results

We pre-processed the documents and transformed them into a numerical, vectorized form by training a Doc2vec model on our data. We inferred new vectors for unseen verses/documents from the test set. We tried different configurations for the hyperparameters of the Doc2vec model, we then trained the classifier with the different versions of the embeddings. Using 80% of the data set to train doc2vec classifier for the Quran verses classification, we achieved 68% accuracy, and 56% F1-score; using the distributed bag of words. We have noticed that changing the vector size did not have a big impact on the classifier performance, but with more training data, the accuracy rises to 70%, and F1- score to 60%. Table 3 shows the classifier performance results using different settings of the model: Distributed Bag-of-words (PV-DBOW) and Distributed Memory (PV-DM).

4.2 Categories Similarity Results

To inspect relationships between the verses numerically, we calculated the cosine distances between their inferred vectors from the trained Doc2vec model. We used this information to calculate the average similarity scores and the mean difference for each category. We want to know how much more similar the same-category documents are to each other than to documents from other categories. Therefore, we created sets of verses pairs for all categories. More precisely, given our fifteen categories which we denoted by $C_1, \ldots, C_{15}$, where each category is a set of verses. Hence, we derived 15 average similarities per each category; one for same-category documents and 14 for cross-category documents. Finally, we calculated the mean similarity differences between the cross-category average similarities and the same-category average similarity. A higher mean difference implies that the model is able to identify documents in one category are more distinct from those in other categories. We experimented with the two architectures of Doc2vec model.

Train set: 80% / Test set: 20%			
Doc2vec model	Vector size	Testing Accuracy	Testing F1score
PV-DBOW	50	0.68	0.56
	100	0.68	0.55
PV-DM	50	0.64	0.54
	100	0.64	0.54
Train set: 90% / Test set: 10%			
Doc2vec model	Vector size	Testing Accuracy	Testing F1score
PV-DBOW	50	0.70	0.60
	100	0.70	0.58
PV-DM	50	0.64	0.55
	100	0.64	0.54

Table 3: Classification Performance Results

Here, we show the results of PV-DBOW architecture, with vector size of 50. We didn't include the results using vectors size of 100 as no impact were noticed. The result of the evaluation can be summarized as in Table 4.

5 Conclusion

We used NLP combined with ML to classify the verses of the Quran into 15 predefined classes. The semantics of the verses were captured using Doc2vec embeddings that were used to group similar documents. Our model achieved a classification accuracy of 68% and 56% F1 score. The results confirmed that the classifier scored higher accuracy with the distributed bags of words architecture of the Doc2vec model. Next, we evaluated our model by examining the semantic similarity of the Quranic verses. Derived classes showed relatively high average similarity for some classes using the distributed bags of words architecture. The three top classes that achieved higher average same-category similarity and mean-difference are Faith, Pillars of Islam, and Religions. The three classes scored top values consistently for both metrics with different runs (500, 700, 900, 1100 test samples). The two metrics are not relatively high for some classes. We contribute that to some classes' documents similar to those of another class. Besides, some verses in the Quran discuss more than one concept/ topic. The uniqueness and complexity of the Quran language could also be a significant reason reflected in our results. Table 5 shows an example of an instance where a verse belongs to different classes/ topics. The results confirmed that the class "Faith" has achieved the highest average similarity and mean difference.

In the future, we may incorporate subtopic chains from Qurany corpus, and we may consider creating unique classifications of Quran verses using existing knowledge resources, and test our model using them. We may consider other approaches for computing the semantic similarity, investigate their performance, and how they compare to our approach.

English Category	(Mean Difference, Same-category Avg. Similarity)
Pillars of Islam	(11%, 15%)
Faith	(21%, 26%)
Man, and The Moral Relations	(2%, 0.74%)
Stories and History	(5.4%, 5.6%)
The Holy Quran	(2.8%, 3.7%)
The work	(2.9%, 6.6%)
Man, and The Social Relations	(3.8%, 7.8%)
Jihad	(0.92%, 0.16%)
Science and Art	(7.2%, 5.3%)
Religions	(13%, 19%)
The Call for Allah	(3.2%, 6.5%)
Trade, Agriculture, Industry and Hunting	(8.6%, 8.7%)
Judicial Relationships	(4.5%, 8.2%)
Organizing Financial Relationships	(2.6%, 4.9%)
General and Political Relationships	(1.5%, 2.7%)

Table 4: Evaluation Results using PV-DBOW, Vector-size =50, # of test samples = 1100.

Verse	**Al-Nahl / سورة النحل (16, 94)**
Arabic Verse	**"وَلاَ تَتَّخِذُواْ أَيْمَانَكُمْ دَخَلاً بَيْنَكُمْ فَتَزِلَّ قَدَمٌ بَعْدَ ثُبُوتِهَا وَتَذُوقُواْ الْسُّوءَ بِمَا صَدَدتُّمْ عَن سَبِيلِ اللّهِ وَلَكُمْ عَذَابٌ عَظِيمٌ"**
English Translation	**And make not your oaths a means of deceit between you, lest a foot should slip after its stability, and you should taste evil because you hinder (men) from Allah's way and grievous chastisement be your (lot).**
Topic	**-Judicial Relationships -Jihad**

Table 5: An example of an instance where a verse belongs to different classes/ topics.

Acknowledgments

The first author is supported by a PhD scholarship from the Ministry of Higher Education, Saudi Arabia. The author is grateful for the support from Jouf University for sponsoring her research.

References

Abu A. Al-Khair, and M. Kabbani. "Koran teacher intonation." The Tunisian Company for Distribution, 2003.

Azman Ta'a, Mohd Syazwan Abdullah, Abdul Bashah Mat Ali, and Muhammad Ahmad. "Themes-based classification for Al-Quran knowledge ontology." In 2014 International Conference on Information and Communication Technology Convergence (ICTC), pp. 89-94. IEEE, 2014.

Menwa Alshammeri, Eric Atwell, and Mhd Ammar Alsalka. "Quranic Topic Modelling Using Paragraph Vectors." Proceedings of SAI Intelligent Systems Conference. Springer, Cham, 2020.

Mohammed Akour, Izzat Alsmadi, and Iyad Alazzam. "MQVC: Measuring quranic verses similarity and sura classification using N-gram." 2014.

Mohammed Naji Al-Kabi, Ghassan Kanaan, Riyad Al-Shalabi, Khalid Nahar, and Basel Bani-Ismail. "Statistical classifier of the holy Quran verses (Fatiha and Yaseen chapters)." Journal of Applied Sciences 5, no. 3 (2005): 580-583.

Mohammed Naji Al-Kabi, Belal M. Abu Ata, Heider A. Wahsheh, and Izzat M. Alsmadi. "A topical classification of Quranic Arabic text." NOORIC 2013: Taibah University International Conference on Advances in Information Technology for the Holy Quran and Its Sciences, 2013.

Noorhan Hassan Abbas. "Quran'search for a concept'tool and website." PhD diss., University of Leeds (School of Computing), 2009.

Suhaib Kh. Hamed, Mohd Juzaiddin Ab Aziz. "Classification of holy quran translation using neural network technique." Journal of Engineering and Applied Sciences 13, no. 12 (2018): 4468-4475.

ABB-BERT: A BERT model for disambiguating abbreviations and contractions

Prateek Kacker[1][*], **Andi Cupallari**[1], **Aswin Gridhar Subramanian**[2][†] and **Nimit Jain**[1]
Novartis Pharmaceuticals, NJ, USA[1]
School of Informatics, University of Edinburgh[2]

Abstract

Abbreviations and contractions are commonly found in text across different domains. For example, doctors' notes contain many contractions that can be personalized based on their choices. Existing spelling correction models are not suitable to handle expansions because of many reductions of characters in words. In this work, we propose ABB-BERT, a BERT-based model, which deals with an ambiguous language containing abbreviations and contractions. ABB-BERT can rank them from thousands of options and is designed for scale. It is trained on Wikipedia text, and the algorithm allows it to be fine-tuned with little compute to get better performance for a domain or person. We are publicly releasing the training dataset for abbreviations and contractions derived from Wikipedia.

1 Introduction

We use abbreviations and contractions (called ”short forms” in this paper) while quickly typing on digital apps. They are used to save time or effort in typing and may be unique to us; therefore, sometimes, only we can understand them. There is no reliable dictionary of short forms to be referred because it can be specific to a context or a person. The short forms often have multiple meanings depending on the domain or the person. In Table 1, consider the sentence *”The doctor saw an AS cd at tl”* written in a notepad by sales representative at pharmaceutical company or local news reporter in Las Vegas. It may be a shorthand for *”The doctor saw an Ankylosing Spondylitis candidate at trial”* for the sales representative or *”The doctor saw an Ace of Spade card at the table”* for the news reporter. Applying downstream NLP AI Algorithms to this shorthand text gives poor performance because they have not been trained on personalized shorthand text. To build better AI systems, we should expand the short forms in the sentences for the domain or the user before using it downstream. Since there is no correct answer for expansions and numerous right choices based on the domain and the user, ranking is the better way to handle short forms text in real-world AI applications.

Text notes	
Notes 1	The doctor saw AS cd at tl
Notes 2	The doctor saw AS cd at tl
Ground Truth	
Notes 1	The doctor saw Ankylosing Spondylitis candidate at trial
Notes 2	The doctor saw Ace of Spades card at the table
ABB-BERT input	
Notes 1	The doctor saw at *[ABB]* *[ABB]* at *[ABB]*
Notes 2	The doctor saw at *[ABB]* *[ABB]* at *[ABB]*
ABB-BERT outputs (sorted list on rank)	
Notes 1	*[ABB]*- [Ankylosing Spondylitis, ...] *[ABB]*- [candidate, ...] *[ABB]*-[trial,...]
Notes 2	*[ABB]*-[Ace of Spades,...] *[ABB]*-[card,...] *[ABB]*-[table,...]

Table 1: Text notes made by one can be ambiguous for others. **Notes 1** was written by pharmaceutical sales, and **Notes 2** was written by local news in Las Vegas. **ABB-BERT** can suggest the best replacement using a fine-tuned model for a domain

The definition of abbreviation is simple, and everyone understands it. For example, *USA* stands for the United States of America, or *MS* stands for Multiple Sclerosis. On the other hand, a contraction is a misspelling or shortening of any word, such as *dr, drs, dctr* etc., for a doctor or *ptnt, pnt, pt* etc., for a patient. Current spelling correction models fail to capture the correct form for all possible scenarios

[*] Correspondence to: prateek.kacker@novartis.com
[†] Part of work was done during employment at Novartis

because of the many reductions of characters in short forms.

Large NLU language models like BERT (Devlin et al., 2019a), RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2020), or DeBERTa (He et al., 2020) are trained on normalized data from different domains but not with personalized or domain-specific short forms and hence reduce the model performance in downstream NLP tasks. For example, in a classification task, the contractions or abbreviation might be critical in determining the class and can lead to a wrong classification (false positive or false negative). To solve this problem, ABB-BERT can normalize the text by ranking the options to find the best choice for abbreviation or contraction, leading to better downstream performance.

In the past, much work has been done on normalizing text data. Misspellings (simple spelling mistakes) have been handled well by AI models. Recent work by Tan et al. (2020) introduced TNT, a model that was developed to learn language representation by training transformers to reconstruct text from operation types typically seen in text manipulation, which they show is a potential approach to misspelling correction. Another AI algorithm Neuspell (Jayanthi et al., 2020) is a spelling correction toolkit that captures the context around the misspelled words. We have noticed that misspelling AI models do not perform well with contractions because of loss in information due to contraction and the number of possible variations for the right choice based on domain.

Kreuzthaler et al. (2016) introduced a data-driven statistical approach and a dictionary-based method for the task of abbreviation detection. They show some success of these approaches; however, as their approach depends on a dictionary with a limited number of entries, it cannot be scaled or extended to other domains. Joopudi et al. (2018) trained Convolutional Neural Network (CNN) models to disambiguate abbreviation senses and found a 1–4 percentage points higher performance for CNN compared to Support Vector Machines. These results were robust across different datasets.

Li et al. (2019) showed that topic modeling combined with attention networks could help get better results on abbreviation disambiguation because topics provide the context for the neural networks. To improve the performance on bio-medical data, Jin et al. (2019b) utilized pre-trained model BioELMO (Jin et al., 2019a) which gets better-contextualized features of words. Then the features are fed into abbreviation-specific bidirectional LSTMs where the hidden states of the ambiguous abbreviations are used to assign the exact definitions. Recently, Pan et al. (2021) proved that BERT-based algorithms combined with training strategies like dynamic negative sample selection and adversarial training are very effective in Scientific AI domain acronym disambiguation datasets (SciAD) (Veyseh et al., 2020).

Algorithm 1 *contraction*

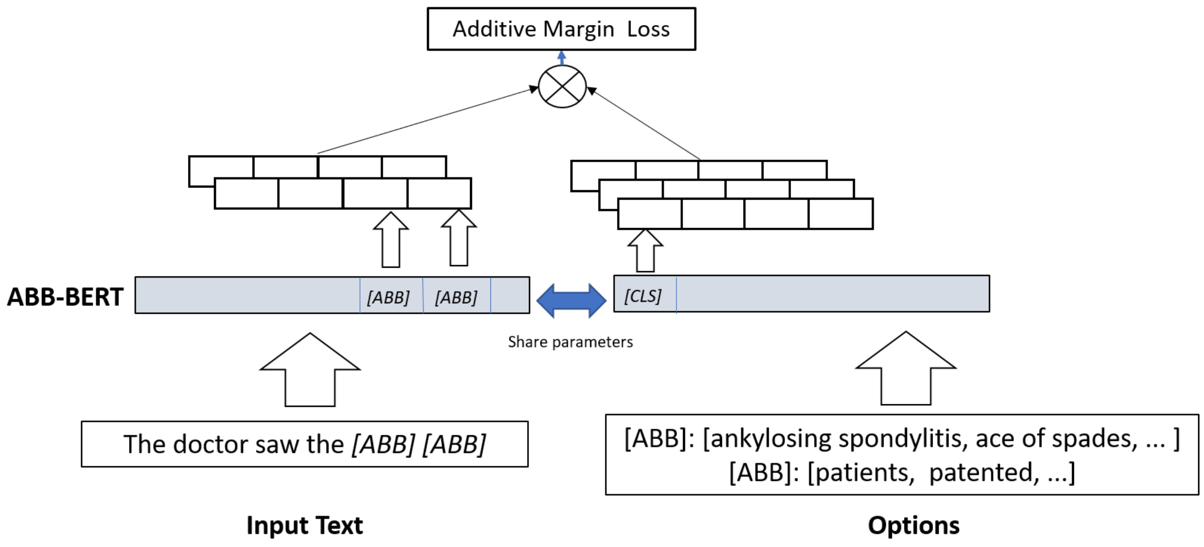
```
Input: word
Output:List of possible contractions
1: Remove any other characters except $a-z$,$A-Z$ and lower case the word
2: Remove all the vowels $a, e, i, o, u$
3: Select all characters except $1^{st}$ character
4: Find all possible combinations of selected characters without changing order
5: Append the first character to each item in the list
6: Return list
```

Algorithm 2 *abbreviation*

```
Input:sentence
Output:List of tuples (Abbreviations,expansions)
1: Identify capitalized word in sentence and their location
2: Identify capitalized word sequences with length two or more
3: If two sequences are seperated by prepositions or conjuctions then connect them to form a sequence
4: Create a list of tuples (initals of uppercase words in sequence, sequence)
5: Return list
```

The contribution of this paper is threefold. First, we propose ABB-BERT which uses a ranking algorithm by combining BERT (Devlin et al., 2019b) and architecture by LaBSE in Feng et al. (2020) on short forms options based on context. We introduce a new token *[ABB]* that replaces all short forms. Second, we show that ABB-BERT is a practical and scalable way to deal with un-normalized text across domains. Third, we publicly release the dataset and code [1] for ABB-BERT for future work.

[1]Dataset and code available at https://github.com/prateek-kacker/ABB-BERT



Additive Margin Loss
ABB-BERT
[ABB]
[ABB]
[CLS]
Share parameters
The doctor saw the [ABB] [ABB]
[ABB]: [ankylosing spondylitis, ace of spades, ...]
[ABB]: [patients, patented, ...]
Input Text
Options

Figure 1: Graphical representation of the training and inference on ABB-BERT. The sentence written by the sales rep is *"The doctor saw the AS ptnt"* but for training, the sentence is modified to *"The doctor saw the [ABB] [ABB]"*. The model is trained to minimize the additive softmax loss of *[ABB]* corresponding to *AS* and *ptnts*. The ground truth for the above example is *"The doctor saw the Ankylosing Spondylitis patient"*. During inference, ABB-BERT ranks the several options given per *[ABB]*. ABB-BERT can be fine-tuned to improve the performance of a domain.

Key	**Value**	**Number of choices**
ptnt	patient, patent, potent, potential ...	4736
dctr	doctor, director, documentary, declaratory ...	2555
tl	table, trial, tool, tuberculosis ...	81236

Table 2: Selected examples of key-value pairs in *dict_cont*.

Key	**Value**	**Number of choices**
as	Ankylosing Spondylitis, Ace of Spades, Astronomy and Space ...	89119
acl	Association for Computational Linguistics, Avant Co. Ltd., Albany Club in London ...	2259
usa	United States of America, Urban Songwriter Award, Ultimate Sports Adventure ...	1608

Table 3: Selected examples of key-value pairs in *dict_abb*.

2 ABB-BERT

The goal of the algorithm is to rank the options for short forms. As shown in 1, the input sentence X may contain one or more short forms. We assume that short forms' location in the sentence and the character composition is known for training purpose. A typical example of sentence with short form can be seen in Table 4. We substitute the short forms with $[ABB]$ and the algorithm uses character composition to get several options for short forms, create embeddings and calculate scores to rank each option.

Original Sentence	**With contractions and abbreviations**
When I got to the house, **Mrs. Everett**, the **housekeeper**, told me that Hermione was in her room, watching her maid pack.	**WI** got to the house, **ME**, the **hs** told me that Hermione was in her room, watching her maid pack.
The Sydney area has been **inhabited** by **indigenous** Australians for at least 30,000 years.	The Sydney area has been **id** by **ig** Australians for at least 30,000 years
Bosnian **claims** of Serbian annexation attempts in 1993 were brought to the **World Court**.	Bosnian **cs** of Serbian annexation attempts in 1993 were brought to the **wc**.

Table 4: Selected examples of GLUE Benchmark datasets. They have been manually edited to create training data for ABB-BERT

2.1 Lookup Tables for Options

ABB-BERT ranks options based on thousands of choices for expansions for short forms present in the lookup tables *dict_cont* and *dict_abb*. To create these tables, English Wikipedia has been parsed for words for contractions and full forms for abbreviations using the rule-based methods in Algorithm 1 and Algorithm 2, respectively. After observing thousands of short forms used in real-world datasets, these rules were created, which we cannot share publicly. Using these rules, we created key-value pairs for lookup tables, *dict_cont* and *dict_abb*, from words and abbreviations extracted from English Wikipedia. Given an abbreviation or contraction, these lookup tables list words that can be the possible expansion. We can see the output of Table 2 and Table 3, and this list can be huge; hence scalability is crucial for ABB-BERT.

2.2 Model

ABB-BERT is based on the BERT architecture. In order to make it lightweight, it is pre-trained on an uncased BERT base model. ABB-BERT requires that every contraction and abbreviation be replaced with *[ABB]* token. Since *[ABB]* is not present in the default vocabulary of BERT, the vocabulary has to be modified to include this special token. Consider a sentence X in dataset D. After the tokenization of X, a sequence of tokens $(x_1, x_2, ...x_n)$ is generated. In this setup, x_1 is the *[CLS]* token for every sentence. We have already replaced the short forms with the *[ABB]* tokens hence we know their exact locations. For simplicity, let $(x_a, x_b, ...)$ represent the *[ABB]* token corresponding to indices $I = (a, b...)$. The BERT output $\boldsymbol{z_1}, \boldsymbol{z_2}, ..., \boldsymbol{z_n}$ corresponding to each token $x_1, x_2, .., x_n$ can be represented as

$$\boldsymbol{z_1}, \boldsymbol{z_2}, ..., \boldsymbol{z_n} = BERT(x_1, x_2, ..., x_n) \tag{1}$$

On top of the BERT model, there is a feed-forward neural network $f(.)$. The output vectors from BERT, $\boldsymbol{z_i}$, go through this neural network such that $\boldsymbol{y_i} = f(\boldsymbol{z_i})$. The final combined representation of the output is

$$\boldsymbol{y_1}, \boldsymbol{y_2}, ..., \boldsymbol{y_n} = \boldsymbol{ABB\_BERT}(x_1, x_2, ..., x_n) \tag{2}$$

$\boldsymbol{y_1}$ is the corresponding output for always the token representing from *[CLS]* and $\boldsymbol{y_a}, \boldsymbol{y_b}$... for *[ABB]* because of the indices I.

We do similar exercise for short forms. Each short form at *[ABB]* can have thousands of options and can be found from *dict_cont* and *dict_abb* tables. For location a, the options are denoted by $\boldsymbol{S_a}$ which is list of options $[S_a^1, S_a^2, ..., S_a^{o_a}]$ and length of the list is denoted by o_a. The tokenizer converts S_a^1, the first option to $(s_a^{1,1}, s_a^{1,2}, ..., s_a^{1,n})$ and similarly for other options $S_a^2, ..., S_a^{o_a}$. Similar to X, every option S_a is propogated through $\boldsymbol{ABB\_BERT}$. The output is represented for S_a^1 is represented as:

$$(\boldsymbol{t_a^{1,1}}, \boldsymbol{t_a^{1,2}}, ..., \boldsymbol{t_a^{1,n}}) = \boldsymbol{ABB\_BERT}(s_a^{1,1}, s_a^{1,2}, ..., s_a^{1,n}) \tag{3}$$

The equation 3 is applied to other options in $\boldsymbol{S_a}$ also. For options, there will not be any *[ABB]*. *[CLS]* is the first and the only relevant token hence the notations can be simplified by dropping the location information. For instance, $s_a^{1,1}$ to s_a^1, $s_a^{2,1}$ to s_a^2 etc and similarly for $\boldsymbol{t_a^{1,1}}$ to $\boldsymbol{t_a^1}$, $\boldsymbol{t_a^{2,1}}$ to $\boldsymbol{t_a^2}$ etc. The algorithm uses additive margin softmax loss, discussed in the next section, to rank the outcomes.

2.3 Dual Encoder with Additive Margin Softmax Loss

The architecture of ABB-BERT with additive margin softmax loss is shown in figure 1. The architecture is similar to the one used by Feng et al. (2020). We use dual-encoders which feeds a scoring function and determines the rank of the alternatives based on the cosine similarity measure, and hence such models are well suited for ranking problems. We use the additive margin softmax loss function introduced in Wang et al. (2018b). Later on, Yang et al. (2019) used a slightly modified version of this loss, and Feng et al. (2020) used it in their language-agnostic LABSE model.

For this paper, the short forms disambiguation problem is modeled as a ranking problem to find the best option S_a for short form at index a in sentence X where S_a is one of the alternatives in $[S_a^1, S_a^2, ..., S_a^{o_a}]$. The ranking of the options is evaluated by the cosine similarity score ϕ . For ABB-BERT, ϕ scores for all the options at location a, is calculated by calculating cosine similarity ϕ between $\boldsymbol{y_a}$ and $\boldsymbol{t_a^1}, \boldsymbol{t_a^2}, ..., \boldsymbol{t_a^{o_a}}$ for options $S_a^1, S_a^2, ..., S_a^{o_a}$. Ranking of the options at location a is done by sorting ϕ scores.

To train the algorithm, we find the conditional probability $P(S|X)$ for options and for all locations. For example, S_a^1, the first option at location a will have $P(S_a^1|X)$ as:

$$P(S_a^1|X) = \frac{e^{\phi(t_a^1, y_a)}}{\sum_{i=1}^{o_a} e^{\phi(t_a^i, y_a)}} \tag{4}$$

This can be extrapolated to other options and other locations. For training purposes, for each location, the first option is the ground truth option.

	Metrics	**Results A**	**Results B**	**Results C**
COLA	Matthews corr.	52.6	22.8	**48.1**
SST2	acc	93.6	20.4	**92.9**
STS-B	Pearson/ Spearman corr	84.9/83.4	62.5/61.2	**75.0/73.8**
QQP	acc./F1	71.6/89.2	54.5/84.9	**65.4/88.3**
MNLI Matched	acc.	84.5	71.3	**77.9**
MNLI Mis-matched	acc.	83.4	72.0	**77.1**
MRPC	acc./F1	86.6/81.6	**79.8/75.4**	75.9/71.5
QNLI	acc.	90.9	**83.4**	82.6
RTE	acc.	64.4	56.7	**57.8**
WNLI	acc.	57.5	**61.0**	58.2

Table 5: Results of inference of downstream task trained on a single BERT-base-uncased model on GLUE Dataset on the respective tasks. Results A are obtained on the original test datasets. Results B are obtained on test datasets manually edited by introducing short forms. Results C are obtained on the test datasets improved by ABB-BERT by selecting 1^{st} option

Additive margin softmax extends the cosine similarity ϕ by introducing a large margin, m, only around correct option. The margin improves the separation between the correct option and other options. Moreover, we scale the cosine values using a hyper-parameter s in the equation 5. We select a large value, which accelerates and stabilizes the optimization (see (Wang et al., 2018b)). Equation 5 represents the loss function and is optimized during training.

Substituting for the new scoring functions, the objective loss function for single sentence X becomes:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=a,b,..}^{I} \sum_{o=1}^{n_i} \frac{e^{s(\phi(t_i^o, y_i)-m)}}{\sum_{k=1}^{n_i} e^{s(\phi(t_i^k, y_i)-m)}} \tag{5}$$

where

$$m = \begin{cases} 1 \geq m \geq \text{-}1 & o\text{=1 or } k\text{=1} \\ 0 & \text{otherwise} \end{cases}$$

$$s \gg 1$$

2.4 Scalable and personalized ABB-BERT

ABB-BERT might have to work real-time during inference in some applications to generate options for downstream tasks, though forward pass through BERT over thousands of alternatives can be expensive and time-consuming. Once ABB-BERT gets deployed, it is expected to get better in ranking for a domain, a user, or group of users with new annotations and training runs; hence there should be a personalization phase equivalent to finetuning the model for a person or a domain. In the *personalization phase*, it is not computationally possible to perform a forward pass on an entire list of options or retrain ABB-BERT again as the inference may be on a device with limited compute. To prepare for the *personalization phase* and to reduce the inference time, *dict_cont* and *dict_abb* is parsed for expansions for all possible options and ABB-

	% Correct outcomes (longest contr.)	**% Correct outcomes (short contr.)**
Wikipedia	63.7	11.2
Covid Dataset	61.7	11.0
Apps Review	54.4	0.0
US Bill	58.9	0.67
ECTHR	70.0	13.9

Table 6: Neuspell results on test sets of different domains on contractions. The model performs well with the longest contraction because it has the most number of words. The performance of Neuspell on abbreviations was close to 0 for all datasets.

BERT embeddings $\boldsymbol{t_i}$ are stored in a lookup table *dict_embed* for each expansion. The parameters for ABB-BERT and the table *dict_embed* are then frozen. ABB-BERT is modified by adding a single feed forward layer $g(.)$ parametrized by θ such that for embeddings y_i for sentences from equation 2 and embeddings $\boldsymbol{t^o_x}$ for options from equation 3, is modified. Training is done only on layer $g(.)$ to reduce training time.

$$\boldsymbol{u^o_i} = g(\boldsymbol{t^o_i}, \theta)$$

and $i \in I$ and for input sentences $\boldsymbol{y_i}$

$$\boldsymbol{z_i} = g(\boldsymbol{y_i}, \theta)$$

Here $\boldsymbol{u^o_i}$ and $\boldsymbol{z_i}$ replaces $\boldsymbol{t^o_i}$ and $\boldsymbol{y_i}$ respectively in equation 5. The model parameters are initialized by θ_0 such that $x = g(x, \theta_0)$. During the personalization phase, only parameters θ are trained which is not computationally expensive and can be done on the device. During inference, ABB-BERT does a forward pass only for input sentence $\boldsymbol{y_i}$. ABB-BERT does not need to do a forward pass on options and it can get embeddings $\boldsymbol{t^o_i}$ directly from *dict_embed* and a forward pass with parameters θ.

3 Experimental Setup

3.1 Data Preparation and Pre-training ABB-BERT

Training of ABB-BERT requires significant preparation of train, test, and validation data. We have taken English Wikipedia and extracted random sentences for datasets. We know that Wikipedia does not have contractions as it is a very clean dataset. Hence we had to create the datasets manually for ABB-BERT based on short forms in algorithm 1 and algorithm 2 respectively. For contractions, 15% of words in a sentence are selected at random. Using algorithm 1, a random contraction is selected to get options from *dict_cont*. Train, test and validation datasets contains >1M, >100K , >100k *[ABB]* tokens respectively. The ground truth, which is the correct expansion, is always the first word of the options in training, test, and validation datasets. In all the datasets, we have only 50 options per *[ABB]*. In real-world scenarios, there will be thousands of options. Pre-training of ABB-BERT was done on NVIDIA K80 GPUs for a week on Wikipedia training data with Adam optimizer and a lr equal to $5e - 06$. After hyper parameter optimization, m was chosen to be 0.8, and s was 30.

3.2 Results

Each sentence in ABB-BERT can have multiple *[ABB]* and performance is calculated at each location at $I = (a, b, ...)$ There are two metrics relevant to this experiment. First is the average of rank (R) of the correct ground truth option, and second is average of difference (Dif) between cosine value (ϕ) of input sentence *[ABB]* & correct option which is the first option in training data and average cosine value of the input sentence $[ABB]$ & rest of the options

$$Dif = \phi(\boldsymbol{t^1_a}, \boldsymbol{y_i}) - (\sum_{n=2}^{o_a} \phi(\boldsymbol{t^n_a}, \boldsymbol{y_i}))/(o_a - 1) \quad (6)$$

We understand that the best average rank of the model outcome on the test set is 1, and Dif should be close to m on average. The larger the value of Dif, the better ABB-BERT is in predicting the outcome.

In the first experiment, we evaluate the impact of short forms on any downstream task. In order to model the impact, we took GLUE Benchmark (Wang et al., 2018a) tasks as a downstream task. Table 5 column **Results A** show the performance of the BERT-base-uncased model on each task without any changes to test data. We manually introduced short forms in test sets of each task using techniques mentioned in section 3.1. There is a marked reduction of performance in most datasets, as shown in table 5 in column **Results B**. Then we corrected each test set with ABB-BERT predictions selecting only the 1^{st} rank option from 50 options. The performance of the new test set is shown in the table 5 in column **Results C**. In the second experiment, we wanted to measure the model performance improvement after the personalization phase. Hence we tested it out on three domain datasets which were bio-medical, legal, and reviews datasets. For the biomedical domain, the Covid-19 QA dataset by Möller et al. (2020) was used. For the legal domain, US Legislation Corpus by Kornilova and Eidelman (2019) and European Court for Human Rights (ECTHR) database by Chalkidis et al. (2019) was used. For the technical domain, the Android Applications User Review dataset by Grano et al. (2017) was used. Sentences from paragraphs were extracted for train, validation, and test datasets. The lookup tables *dict_cont* and *dict_abb* were used to create short forms for all the datasets and parameters θ of $g(.)$

	Pre-personalization			Post-personalization					
	Avg. Rank over 50 options	[ABB] count	Avg. Diff	%(Top 3 ranks) in test set	Avg. Rank	Avg. Rank improvement	count of [ABB] Rank increase	Avg. Rank decrease	count of [ABB] Rank decrease
Wikipedia (Pre-training)	1.45	125079	0.67	-	-	-	-	-	-
Covid Dataset	1.58	16218	0.61	95.7	1.57	2	174	1.18	130
Application Review	1.42	9150	0.67	97.7	1.32	8.89	140	1.6	164
US Bill	1.51	29470	0.64	96.5	1.46	5.17	396	1.49	332
ECTHR	1.26	22295	0.67	98.5	1.25	3.24	144	1.2	254

Table 7: ABB-BERT performance on different domain data pre and post personalization phase. In pre-personalization phase, ABB-BERT was used without any domain training. The results are for short forms identified together in a sentence. m is 0.8 and Avg Diff is very close to it. The average rank without training is very high leaving little scope of big improvement. However there is improvement seen in rank of the correct option in post-personalization phase

were trained keeping $\boldsymbol{ABB\_BERT}$ parameters static for this experiment. The number of options for every $[ABB]$ was 50 for every dataset.

ABB-BERT performance results on a test set are shown in table 7. Without any training of parameter θ, ABB-BERT does very well in the pre-personalization phase. The performance on the test set in post-personalization gets better though not noticeable because the average rank is close to 1 in pre-personalization phase.

In the third experiment, we wanted to compare our work with existing work. We could not find the exact equivalent for this work, but we still decided to baseline this work for contractions using NeuSpell by Jayanthi et al. (2020). Neuspell is an excellent algorithm for misspellings, but when exposed to contractions, it makes many mistakes. Results of the baseline can be found in table 6. As expected, NeuSpell does well for lengthiest contractions than shortest contractions. The performance of Neuspell was close to 0 for all the abbreviation datasets. Hence, the results are not shown in the table 6.

For abbreviations, we tested out the algorithm on the SciAD dataset from Veyseh et al. (2020). ABB-BERT, without training, gives an average rank of 1.76, which is lower compared to the best model by Jin et al. (2019b). It is because the loss function of the algorithm is designed for ranking on large number of options. However, the performance improves after the personalization phase, with an average ranking of 1.55.

3.3 Visualizing ABB-BERT results

In the datasets, ABB-BERT is given only 50 options. It does a great job in predictions with more than 90% performance for the top 3 choices. If we look at the results, we see that most of the time, the one of top choices can make a correct substitution for $[ABB]$ in a sentence. Table 8 shows the options that scored high and make much sense. It shows that model can learn grammar and understands language well. However, the model does not consider commonsense or missing context in ranking. Table 9 shows where the model makes mistakes in predictions because of inherent challenges in this task.

4 Conclusions and future work

In this work, we propose ABB-BERT for abbreviation and contraction disambiguation. ABB-BERT tackles both of these irregularities in the text simultaneously and has the advantage that it takes into account the context in the sentence when ranking the possible alternatives. We designed it on Wikipedia and tested it on domain data also. The model may have a hard time getting the right options if they are grammatical appropriate based on context but maybe wrong by commonsense. Future work can help improve the model and suggest all *[ABB]* at the same time based on commonsense and missing context like geographical location, history

Original Sentence	**With** $[ABB]$	**top 5 alternatives and cosine scores**
Young redheaded man holding two bicycles near beach.	Young redheaded man holding [ABB] [ABB] near beach.	**ABB1**:(**two**, 0.99), (twag, 0.20), (twili,0.20), (tmfw,0.20), (townian,0.20) **ABB2**:(**bicycles**: 0.86), (berchy: 0.20), (binchy: 0.20), (bakley: 0.20), (besyde: 0.20)
This problem has been **previously** studied for zero-shot object **recognition** but there are several key differences.	This problem has been [ABB] studied for zero-shot object [ABB] but there are several key differences.	**ABB1**:(**previously**, 0.99), (provincial, 0.98), (privateering, 0.2), (pāval, 0.2), (primavera, 0.2) **ABB2**: (**recognition**, 0.99), (recréation, 0.26), (retroactive, 0.21), (rectification, 0.21), (revoluction, 0.20),
a **vivid cinematic** portrait.	a [ABB] [ABB] portrait.	**ABB1**:(**vivid**, 0.99), (vmvs, 0.20), (vhvi, 0.20), (vitruvius, 0.20), (vouvantes, 0.20) **ABB2**: (**cinematic**, 0.99), (christini, 0.20), (ciston, 0.20), (coeffient, 0.20), (clairant, 0.20)

Table 8: Selected examples of GLUE Benchmark datasets. The models made an accurate predictions on the options it was given. The model understands grammar and takes in context in the sentence

Original Sentence	**With** $[ABB]$	**top 5 alternatives and cosine scores**
”It's our judgment that the possible avenues to a peaceful resolution were **not** fully explored at the Tokyo conference,” **U.S. State Department** spokesman **Richard Boucher** said.	” It's our judgment that the possible avenues to a peaceful resolution were [ABB] fully explored at the Tokyo conference,” [ABB] spokesman [ABB] said.	**ABB1**: (**not** ,0.99), (nudator, 0.204), (nafat, 0.204),(ndkt, 0.204), (nonotic, 0.203) **ABB2**:(United States Delegation, 0.99), (Ukrainian Second Division, 0.99),(Ukrainian Soviet Division, 0.99), (Ukrainian Social Democratic, 0.99), (Union of Social Democrats, 0.99), **ABB3**: (road between,0.99), (Rob Bradley, 0.99), (Rosario Blanco, 0.99), (Ralph Barbara,0.99), (Roger Barclay,0.99)
Maude and Dora saw a **train** coming	[ABB] saw a [ABB] coming	**ABB1**:(Mountain Daughter,0.99), (Mo Due, 0.99), (Molino Dam, 0.99), (Mustard Digital, 0.99), (**Maude and Dora**, 0.99) **ABB2**:(**train**, 0.99), (tegne, 0.20), (tanshi, 0.20), (thunderer, 0.20), (trumain, 0.20), (tenelea, 0.20),
Alan J. Konigsberg is related to **Levy Phillips & Konigsberg**	[ABB] [ABB] related to [ABB]	**ABB1**:(All Japan Kick,0.99), (Archbishop John Kemp,0.52), (Arbab Jehangir Khan,0.35), (American John Kendrick,0.3), (Albert James Kingston,0.29) **ABB2**:(is, 0.99), (istd, 0.20), (isthmo, 0.20), (inscs, 0.20), (istres,0.20), **ABB3**:(Lord Palatine of Kyiv, 0.99), (Liverpool Park Keepers, 0.99), (La Palabra Kilometros, 0.99), (Long Pine Key, 0.99), (Lalitha Priya Kamalam,0.91)

Table 9: Selected examples of GLUE Benchmark datasets. The sentences are hard to predict because of the model outcome might be correct grammatically but does not match the ground truth. Is some cases enough information is not provided as input to make a correct prediction

of notes etc.

References

Ilias Chalkidis, Ion Androutsopoulos, and Nikolaos Aletras. 2019. Neural legal judgment prediction in English. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4317–4323, Florence, Italy. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019a. Bert: Pre-training of deep bidirectional transformers for language understanding.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019b. 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.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2020. Language-agnostic bert sentence embedding.

Giovanni Grano, Andrea Di Sorbo, Francesco Mercaldo, Corrado A. Visaggio, Gerardo Canfora, and Sebastiano Panichella. 2017. Android apps and user feedback: a dataset for software evolution and quality improvement. In *Proceedings of the 2nd ACM SIGSOFT International Workshop on App Market Analytics*, pages 8–11.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention.

Sai Muralidhar Jayanthi, Danish Pruthi, and Graham Neubig. 2020. Neuspell: A neural spelling correction toolkit.

Qiao Jin, Bhuwan Dhingra, William Cohen, and Xinghua Lu. 2019a. Probing biomedical embeddings from language models. In *Proceedings of the 3rd Workshop on Evaluating Vector Space Representations for NLP*, pages 82–89, Minneapolis, USA. Association for Computational Linguistics.

Qiao Jin, Jinling Liu, and Xinghua Lu. 2019b. Deep contextualized biomedical abbreviation expansion. In *BioNLP@ACL*.

Venkata Joopudi, Bharath Dandala, and Murthy Devarakonda. 2018. A convolutional route to abbreviation disambiguation in clinical text. *Journal of Biomedical Informatics*, 86:71–78.

Anastassia Kornilova and Vlad Eidelman. 2019. Billsum: A corpus for automatic summarization of us legislation.

Markus Kreuzthaler, Michel Oleynik, Alexander Avian, and Stefan Schulz. 2016. Unsupervised abbreviation detection in clinical narratives. In *Proceedings of the Clinical Natural Language Processing Workshop (ClinicalNLP)*, pages 91–98, Osaka, Japan. The COLING 2016 Organizing Committee.

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 *International Conference on Learning Representations*.

Irene Z Li, Michihiro Yasunaga, Muhammed Yavuz Nuzumlali, C. Caraballo, Shiwani Mahajan, H. Krumholz, and Dragomir R. Radev. 2019. A neural topic-attention model for medical term abbreviation disambiguation. *ArXiv*, abs/1910.14076.

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.

Timo Möller, Anthony Reina, Raghavan Jayakumar, and Malte Pietsch. 2020. Covid-qa: A question answering dataset for covid-19. In *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*.

Chunguang Pan, Bingyan Song, Shengguang Wang, and Zhipeng Luo. 2021. Bert-based acronym disambiguation with multiple training strategies.

Fei Tan, Yifan Hu, Changwei Hu, Keqian Li, and Kevin Yen. 2020. TNT: Text normalization based pre-training of transformers for content moderation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4735–4741, Online. Association for Computational Linguistics.

Amir Pouran Ben Veyseh, Franck Dernoncourt, Quan Hung Tran, and Thien Huu Nguyen. 2020. What does this acronym mean? introducing a new dataset for acronym identification and disambiguation. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 3285–3301. International Committee on Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018a. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Feng Wang, Weiyang Liu, Haijun Liu, and Jian Cheng. 2018b. Additive margin softmax for face verification. *CoRR*, abs/1801.05599.

Yinfei Yang, Gustavo Hernandez Abrego, Steve Yuan, Mandy Guo, Qinlan Shen, Daniel Cer, Yun hsuan Sung, Brian Strope, and Ray Kurzweil. 2019. Improving multilingual sentence embedding using bi-directional dual encoder with additive margin softmax.

Training data reduction for multilingual Spoken Language Understanding systems

Anmol Bansal [*†§], **Anjali Shenoy**[*‡], **Chaitanya P. K.**[‡], **Kay Rottmann**[‡], **Anurag Dwarakanath**[‡]
[‡]Alexa AI, Amazon
[§]IIT Kharagpur
{anshen, kppappu, krrottm, adwaraka}@amazon.com
anmolbansal@iitkgp.ac.in

Abstract

Fine-tuning self-supervised pre-trained language models such as BERT has significantly improved state-of-the-art performance on natural language processing tasks. Similar fine-tuning setups can also be used in commercial large scale Spoken Language Understanding (SLU) systems to perform intent classification and slot tagging on user queries. Fine-tuning such powerful models for use in commercial systems requires large amounts of training data and compute resources to achieve high performance. This paper is a study on the different empirical methods of identifying training data redundancies for the fine tuning paradigm. Particularly, we explore rule based and semantic techniques to reduce data in a multilingual fine tuning setting and report our results on key SLU metrics. Through our experiments, we show that we can achieve on par/better performance on fine-tuning using a reduced data set as compared to a model fine-tuned on the entire data set.

1 Introduction

In recent years, a variety of smart voice assistants such as Apple's Siri, Samsung's Bixby, Google Home, Amazon Echo, Tmall Genie, have been deployed and achieved great success. These voice assistants facilitate goal-oriented dialogues and help users to accomplish their tasks through voice interactions. One component of such spoken language understanding (SLU) systems is Natural Language Understanding (NLU) which aims to extract the intent of the query (intent classification) and semantically parse the user utterance (slot tagging). As an example, if a user requests "play madonna" to the voice assistant, SLU would classify the intent as *PlayMusic* with slot filling of tokens "play" as *Action* and "madonna" as *Artist*.

[*]equal contribution
[†]work done while interning with Alexa AI

As in many other language processing fields, pre-trained language models have seen major success for natural language understanding. Pre-trained language models (Radford and Narasimhan, 2018; Howard and Ruder, 2018; Baevski et al., 2019; Dong et al., 2019) are generic language models learned in a semi-supervised fashion whose underlying large scale knowledge is then leveraged for fine-tuning towards down-stream tasks (Ruder et al., 2019). BERT (Devlin et al., 2019) is one such example of a language model based on the Transformer Network architecture (Vaswani et al., 2017), pre-trained on a corpora of 3300M words extracted from publicly available unannotated data and then fine-tuned on smaller amounts of supervised data for specific tasks, relying on the induced language model structure to facilitate generalization beyond the annotations. It provides powerful and general-purpose linguistic representations, triggering strong improvements and significant advances on a wide range of natural language processing tasks. Chen et al. (2019) also observed the success of BERT to jointly learn intent classification (IC) and slot filling (SF) tasks by leveraging pre-trained representations. Leveraging this joint IC and SF set up to interpret user utterances in commercial SLU systems requires large volume of annotated training data (Ezen-Can, 2020), compute resources (such as GPUs) and model build time to cover the variability of customer utterances. Such resources (human and compute) are not only expensive but also time consuming, unscalable and may not be fully optimized to achieve the same performance.

In this paper, we perform an empirical analysis on identifying subsets of multilingual training data which can achieve on par or better performance as compared to the same model trained

Proceedings of the 18th International Conference on Natural Language Processing, pages 298–306
Silchar, India. December 16 - 19, 2021. ©2021 NLP Association of India (NLPAI)

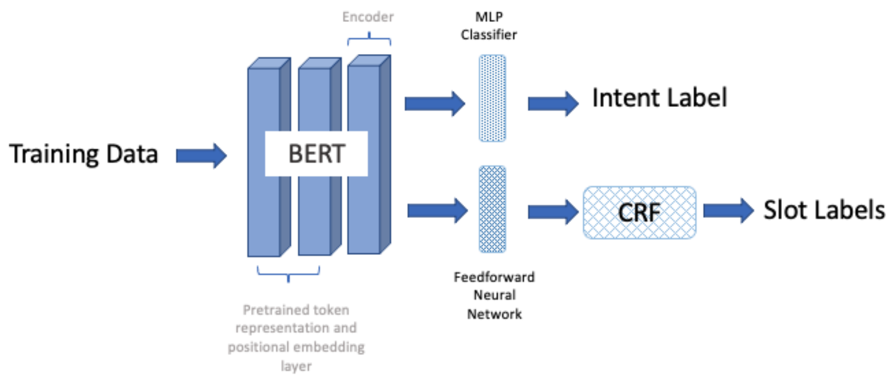

Encoder
MLP Classifier
Intent Label
Training Data
BERT
CRF
Slot Labels
Feedforward Neural Network
Pretrained token representation and positional embedding layer

Figure 1: IC/SF Bert architecture

on the full data-set for the same task. In particular, we fine-tune a Lean-BERT sized model (Conneau et al., 2020), on the IC-SF task for Hindi and English SLU data. Such mutlilingual large scale pre-training is known to effectively promote cross-lingual generalization (Choi et al., 2021; Pires et al., 2019) giving rise to an opportunity to exploit latent space similarities of such languages to identify redundancies in training data for fine-tuning. We experiment with various semantic and rule based data reduction approaches and report fine-tuning performance on key SLU metrics.

2 Related work

Finding the right data reduction technique for BERT fine tuning while maintaining evaluation performance can be considered as a part of two major classes of problems - fine tuning regime in the low resource setting to leverage insights from incorporated best practices, and the few shot classification class of problems where the model is trained using only a few samples from each class. Note that the above two classes of problems are not disjoint and is concurrently explored in this work.

2.1 Low resource fine tuning

A newly discovered approach to fine-tune a transformer based model using low resource data is pre-finetuning, introduced in Aghajanyan et al. (2021). In pre-finetuning, the BERT based model undergoes large-scale multi-task learning between language model pre-training and fine-tuning to encourage learning of representations that generalize better to many different downstream tasks. Pre-finetuning gains are particularly strong in the low resource regime, where there is relatively little labeled data for fine-tuning. Our proposed approaches can be used as an extension on top of pre-finetuning to use the gains of the pre-finetuning and benefit from smaller data-sets during the real finetuning.

Active learning is also a widely popular space involving few shot learning where the number of examples to learn a concept are much lower than that required in a normal supervised learning setting. Grießhaber et al. (2020) explore active learning in conjunction with BERT finetuning in the low resource setting with less than 1000 data points. The method involves using Bayesian approximations of model uncertainty (Gal and Ghahramani, 2016) to efficiently select unannotated data for manual labeling. The method utilizes pool-based active learning to speed up training while keeping the cost of labeling new data constant. They also demonstrate the benefits of freezing layers of the pre-trained language model during fine-tuning to reduce the number of trainable parameters, making it more suitable for low-resource setting. Drawing inspiration from this, we conduct our experiments by initially freezing the input embedding layer and gradually unfreezing it by applying an increasing fraction of the learning rate over the training steps.

Shnarch et al. (2021) introduce a new unsupervised learning layer between pre-training and fine-tuning called the Clustering Layer which helps train BERT on predicting cluster labels and can significantly reduce the demand for labeled examples for topical classification tasks. This technique however affects the overall latency of the model in real time systems and we only wish to consider those techniques which modify the input training data rather than the model itself.

Zhang et al. (2021) explore commonly observed instabilities in few-sample scenarios for fine-tuning BERT. Several factors which were identified as causes of instability were the limited applicability of significant parts of the BERT network for downstream tasks and the prevalent practice of using a pre-determined small number of training iterations. We have leveraged insights from this work and accordingly tuned the various hyper parameters of our model.

2.2 Few shot classification & entity recognition

While few shot and one shot learning techniques are very popular in computer vision for tasks such as image recognition (Koch, 2015), in NLP Lampinen and McClelland (2018) was the first to introduce one-shot and few-shot learning for word embeddings. Geng et al. (2019) explore leveraging the dynamic routing algorithm in meta-learning (Yin, 2020) to simulate the few-shot task and introduce a novel Induction Network to learn generalized class-wise representations. Huang et al. (2020) explore few shot learning for the entity recognition task with meta learning, supervised pre-training (similar to BERT) and self-training to leverage unlabeled in-domain data. Yang and Katiyar (2020) explore entity recognition in the nearest-neighbour paradigm.

The first works in data reduction techniques in Machine Learning (Wilson and Martinez, 2004; Arnaiz-González et al., 2016) are based on instance selection methods broadly classified into two categories. The first is the incremental method which begins with an empty set and the algorithm keeps adding instances to the this subset by analyzing instances in the training set. The decremental method, on the contrary, starts with the original training data set removes those instances that are considered superfluous or unnecessary. We would consider our approach of selecting the subset of data as a decremental method since we start from the original set and proceed to extract a smaller set from it.

Koh and Liang (2020) introduced the concept of influential data instances - those training points which are most responsible for a given prediction - and how to identify them. However, this approach can only be applied to machine learning models trained on convex losses and is also not scalable due to the computationally heavy Hessian matrix multiplication involved. Pruthi et al. (2020) extended this concept to estimate training data influence by tracing its gradient descent. Using first-order approximation for Hessian computation and extending the algorithm to mini-batches, they made this approach scalable and showed results on an image classification task. This is however unexplored in the language processing setting for joint intent classification and slot filling task which is more complex than binary classification.

3 Method

In this paper, we first describe the SLU architecture used for the IC-SF task and the four methods for data reduction.

3.1 SLU model Architecture

We use a common SLU architecture (Chen et al., 2019) for joint intent classification and slot filling, which is depicted in figure 1. It consists of a BERT based encoder, an intent decoder and a slot decoder. The BERT encoder's outputs at sentence and token level are used as inputs for the intent and slot decoders, respectively. The intent decoder is a standard feed-forward network including one standard task specific layer and a softmax layer on top. Meanwhile, the slot decoder uses a CRF layer on top of one task specific layer to leverage the sequential information of slot labels. During the training, the losses of IC (cross-entropy loss) and SF (CRF loss) are optimized jointly with equal weights as in Chen et al. (2019)

3.2 Data reduction approaches

We define terminologies that we use throughout the paper as the following - Let an utterance $u_i \in S$, where S is the set of all utterances in the training data, have an intent $intent_i \in I$ and annotation $a_i \in A$, where I and A is the set of all intents and annotations and $a_i \in A$ is a string of tokens with each token annotated with a slot label. From our previous example, u_i = "play madonna", $intent_i$ = PlayMusic, a_i = "play<Action> madonna<Artist>"

3.2.1 Baseline

In the baseline, we fine-tune model on the entire data set (S, I, A) without any modifications.

3.2.2 Unique

In this method, for an utterance $u_i \in S$, we extract unique utterance annotations filtered using annotation $a_i \in A$ as a key. This helps in uniformly representing variations in SLU data by removing any bias due to frequency of occurrence.

3.2.3 Log N

In Log N reduction, if an utterance $u_i \in S$ has a frequency of occurrence n_i, we downsample the

Domain	Unique	Log	Clustering 70%	Singular Score
Music	-55%	-38%	-30%	-25%
Shopping	-51%	-40%	-30%	-35%
Video	-44%	-24%	-27%	-18%
Notifications	-52%	-36%	-33%	-11%
Weather	-53%	-38%	-20%	-3%

Table 1: Reduction achieved by different techniques.

utterance to have a frequency $\log_2(n_i)$. This maintains the utterance distribution as in the original dataset but reduces the absolute magnitude of the frequency. This way the model learns the original input distribution of the SLU data but the reduced representation helps avoid overfitting the model to the more prevalent classes. We experimented with other variants of Log N subsampling such as k-Log N where $k \in \mathbb{N}$ and would involve scaling the frequency to k times $\log_2(n_i)\ \forall u_i \in S$ but we did not see any significant gains in this approach.

3.2.4 Clustering

The first two methods described above only account for the frequency statistics of the utterance in the training data and is language agnostic. In the clustering approach, we try to reduce the data distribution *semantically*.

The steps in the clustering approach are as follows:

- Identify a subset of intents $I' \subseteq I$ by filter for those where the number of utterances $u_i \in S$ labeled with $intent_i \in I'$ is greater than 1000. This is done so that we do not reduce the data from underrepresented intents.
- Extract unique utterances for all utterances having $intent_i \in I'$ using a_i as the key. Repetitions of utterances in the data will have the same word embedding representation and creates redundancy in the input to the clustering algorithm and a compute resource bottle neck.
- Extract embedding representation $e_i \in \mathbb{R}^m$ having dimensions m for these unique utterances from the max pooling representation of the model's [CLS] token. (Devlin et al., 2019).

$$M_{n\times m} = \begin{bmatrix} e_1 \\ e_2 \\ . \\ . \\ e_n \end{bmatrix}$$

- Condense these unique utterance's embedding representations into a smaller number of dimensions $d < m$ by computing SVD on the input matrix

$$M_{n\times m} = U_{n\times m}\Sigma_{m\times m}V_{m\times n}$$

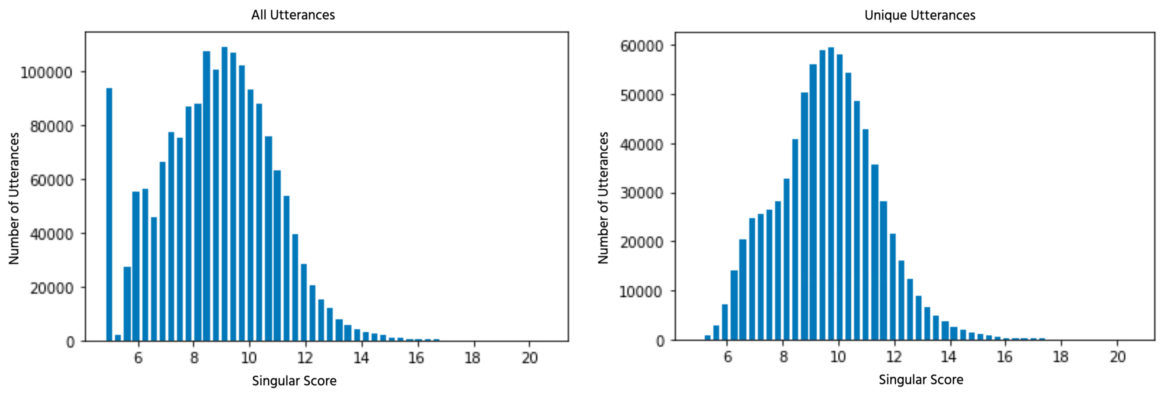
$$M_{n\times m}\bar{V}^T_{m\times d} = U_{n\times m}\bar{\Sigma}_{m\times d}$$

where $\bar{V}^T_{m\times d}$ and $\bar{\Sigma}_{m\times d}$ are simply the first $d < m$ columns of V and Σ.
- Obtain the condensed representation for each unique utterance's data point in the rows of $U_{n\times m}\bar{\Sigma}_{m\times d}$. Note that unlike PCA we do not normalize the input here since it is computationally expensive.
- For each $intent_i \in I'$, perform k-means clustering from the extracted and condensed utterance embedding representations and find the optimal number of clusters K using the Elbow Method.
- Restore the frequency of the clustered utterances to the original frequency as observed in the full dataset S.
- Per cluster $k_i \in K$ where the clustered utterances have their original frequency of utterance, randomly sub-sample 30% of the utterances and use the remaining 70% as the training set.

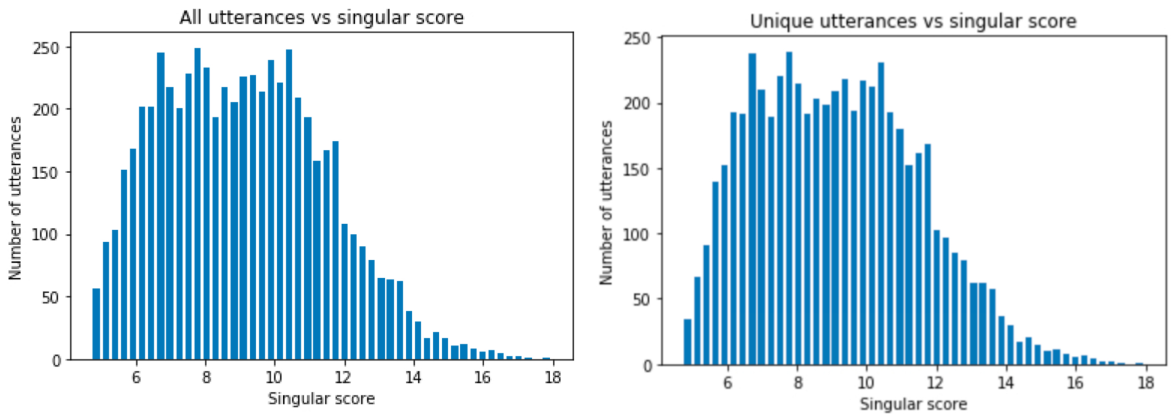
Note that we experimented with choosing a subset by randomly subsampling [10%, 20%, 30%, 50%] of the data and observed that subsampling 30% of the data had the best balance with respect to amount of reduction versus performance drop.

3.2.5 Singular Score

Golub and Reinsch (1971) introduced Singular Value Decomposition (SVD), a technique to factorize the matrix into two unitary matrices and a diagonal matrix. The diagonal values of this matrix are the singular values. This approach has been used extensively in multiple fields since its onset in 1970s such as the work described in (Kabsch,



(a) Our dataset



(b) Multi ATIS++ dataset

Figure 2: Singular Scores Distribution

1978), which uses SVD to compute an ideal rotation matrix for 3-D molecular comparisons. (Walton et al., 2013) explores the decomposition offered by SVD to reduce the degrees of freedom and interpolate the flow problem to lower complexity, with minimal loss in accuracy of representation. In the field of Statistics and Machine Learning, it has been primarily used to achieve dimensionality reduction with minimal loss in information content. One such application in field on Information Retrieval is Latent Semantic Analysis (LSA) (Furnas et al., 1988) where sparse representations of documents were reduced significantly to a few dimension that hold most information and these were used as representations for the original documents.

For this work, we explore using SVD on subsampling data-points instead of subsampling dimensions as in regular applications. As seen in equation 1, M is the embedding matrix with n datapoints and m dimensions per datapoint. We performed experiments with treating data-points analogous to dimensions and subsampling them. However, this wasn't favorable as the datapoints being treated as dimensions for reduction were very large in number and did not have the correlation factor as seen with regular dimensions of embeddings.

$$M_{n\times m} = U_{n\times m}\Sigma_{m\times m}V^{T}_{m\times m} \qquad (1)$$

$$B = MV = U\Sigma \qquad (2)$$

We instead analyse the projection of each utterance on principal axes and formulate a score, which we will refer to as the Singular Score going forward. We use this score value to quantize the dataset into buckets and apply appropriate downsampling methods per bucket, giving up to 25% reduction in the data while also showing improvements in the SemER and Intent Classification metric consistently.

From equation 2 we can see that MV, which is the projection of embedding matrix along principal components is the same as $U\Sigma$. This is because U, V are unitary matrices and $VV^T = V^TV = I$. Each row in this matrix $B = MV$ represents the projection of corresponding input embedding along principal axes. We use absolute sum (i.e L_1 norm, or Manhattan distance from origin) of each row in B as the Singular score corresponding to the data point $u_i \in S$.

$$score_i = \sum_{j=1}^{m} |B_{ij}| \tag{3}$$

We conducted experiments on the representation power of this score and find interesting observations.

Correlation with frequency: As shown in figure 2a we find that there is a correlation between the frequency of an utterance and the score it generates where the lower ranges of scores represent more than 60% of the data. We also performed this experiment on the English and Hindi subset of the Multi-Atis++ data (Xu et al., 2020) to verify our observations. The Multi-Atis++ dataset is an extension of the ATIS (Air Travel Information Services) dataset (Upadhyay et al., 2018) developed to support the research and development of speech understanding systems. This data comprises of 5928 user spoken utterances (4488 English, 1440 Hindi) of which 5621 (94%) utterances are unique. These utterances are based on various hypothetical travel planning scenarios and are obtained by users interacting with a partially or completely automated ATIS system which is then recorded and transcribed. As shown in figure 2b we see that due to the unique utterance composition of Multi-Atis++ Hindi and English subset, the singular score distribution of the graphs remain majorly the same.

Correlation with Sparsity: We also observe that singular scores are a loose indicator of sparsity in the principal axis space as shown in figure 3. A datapoint with higher singular score is observed to have dense representation in its projection along principal axes while an utterance with low singular score has a representation on one or two of the first few axes only. Since the first few principal axes in SVD indicate the spread of the data on those axes, lower singular scores which primarily contain scores in the first few axes belong to those utterances which are common in the data.

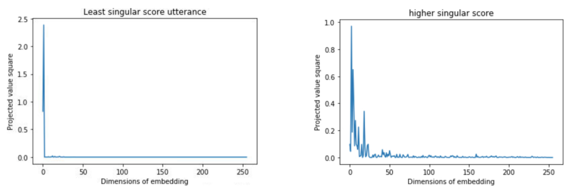


Figure 3: Singular Score Comparison

Compounding these two observations, and based on the pattern we observe in figure 2a we use the Singular Scores to quantize the utterance bins into three and apply different degrees of subsampling to each bucket. For an utterance $u_i \in S$ with frequency n_i and singular score value $score_i$:

- *Head:* Utterances with low singular scores, ($score_i \leq 7.7$) which have a higher degree of repetition. The frequency n_i is reduced to $10 * \log_2(n_i)$
- *Mid:* Utterances with medium singular scores ($7.7 < score_i \leq 10$) has its frequency n_i reduced to $\log_2(n_i)$
- *Tail:* Utterances with high singular scores ($score_i > 10$) which have almost negligible repetitions have their frequency retained.

The amount of reduction achieved by the Unique, Log N, Clustering 70% and Singular Score approaches is summarized in table 1.

4 Experimental Setup

4.1 Data

Since we present approaches with practical applications to real-world SLU modelling systems, we present results on real world data. In particular, use 3 months of data from an internal de-identified data authority and include a random sample from the remainder of the year to account for seasonality in the utterance requests. We use English and Hindi data from five domains, i.e. *Music, Video, Weather, Notifications,* and *Shopping*.

Data statistics are shown in table 3; for each domain, we have atleast 100k training samples of English and Hindi data in equal distribution and use 90% for training and 10% for validation.

	Domain	Unique		Log N		Clustering 70%		Singular Score	
		SemER	IC	SemER	IC	SemER	IC	SemER	IC
English	Music	-6.18	-18.24	-3.77	-14.46	-2.48	-8.83	-3.08	-14.45
	Shopping	-3.52	-3.29	-2.88	-3.84	-4.19	-4.53	-3.79	-4.97
	Video	+9.90	+10.04	+4.40	+4.45	+6.31	+5.70	+3.80	+4.05
	Notifications	-1.10	-0.19	-1.90	-0.62	-1.00	-0.08	-1.15	-2.56
	Weather	-4.40	-9.25	-2.20	-4.29	-5.28	-7.86	-8.05	-17.14
Hindi	Music	-6.60	-24.48	-4.66	-20.74	-2.32	-10.77	-4.51	-20.11
	Shopping	-7.20	-9.71	-4.45	-3.83	-2.23	-3.87	-5.13	-6.63
	Video	+9.54	+10.96	+5.29	+6.62	+1.76	+6.16	-0.00	+6.16
	Notifications	-2.72	-6.21	-2.16	-7.45	-3.47	-13.69	-2.60	-11.20
	Weather	-1.56	-28.21	-0.00	-15.38	-1.12	-25.64	-1.12	-23.08

Table 2: Relative change results

Domain	Intents	Slots
Music	27	103
Shopping	25	45
Video	36	73
Notifications	24	47
Weather	4	18

Table 3: Dataset distribution

4.2 Model parameters

We use an in-house distilled multilingual Lean BERT (Conneau et al., 2020) sized model (`50M`parameters) pre-trained on multiple languages including English and Hindi on a large variety of tasks. We use max-pooling for sentence representation. Each of our decoders, i.e. for IC and slot filling components, have one dense layer of size 128 and 256 with relu and gelu activation each respectively. The dropout values used in IC and SF decoders are 0.1. For optimization, we use Adam optimizer with learning rate $10e^{-4}$ with a step scheduler. We trained our model for 15 epochs with batch size of 64 and gradually unfreeze the initial embedding layer (Howard and Ruder, 2018) over 5000 steps.

4.3 Metrics

We evaluate our models on two standard SLU metrics - Intent Classifcation accuracy (IC) and Semantic Error Rate (SemER) following Gaspers et al. (2018), which jointly measures IC and SlotF1 and is defined as

$$SemER = \frac{\#(slot + intent\_errors)}{\#slots\_in\_reference + 1} \qquad (4)$$

5 Results & Discussion

For each domain, we build four SLU models trained on the combined English and Hindi data, each named after the data reduction approach applied to the training data fed to it: Baseline, Unique, Log N, Clustering 70%, and Singular Score. We report the performance for each model on SemER and IC accuracy metrics in table 2

We break down our results into three categories: discussion on Video domain degradation, performance analysis of various data reduction technique and performance comparison across metrics.

5.1 Video domain degradation

Video domain consistently sees degradation in SemER metric as compared to the baseline model trained on the complete dataset. This is an indicator that subsampling data is not always beneficial and should be leveraged to make decisions on whether the data slice should be subsampled or not. However, degradation was also observed to be the least in the Singular Score approach, with 0% delta for Hindi SemER and the least IC degradation score. The Video domain training data singular scores captures the essence of frequency and semantic variety in training data which the Unique, Log N and Clustering 70% methods individually could not, furthering concreting our belief in the intuition behind these scores.

5.2 Data reduction techniques

The method of uniquing the input data performs well across languages and metrics as compared to the other approaches. However, this is not practical for commercial SLU systems where the natural distribution of utterance weights is determined by its frequency of occurrence. Similarly, for the Log N approach, we see consistent improved performance, yet this approach affects tail frequency utterances which are already under represented. We can perform Log N reduction on only the top few most frequent utterances and generate a uniform representation from the long tail distribution of data, but that would be a scaled version of the unique experiment and we expect results to be pretty similar.

An interesting observation we extract from the results table is that for the Singular Score approach, across most domains, most metric values have the behaviour

$$\text{Sing. Score} \leq min(\text{Log N}, \text{Clust. } 70\%)$$

Singular Score method shows combined improvements from Log N and Clustering 70% indicating that the Singular Score approach factors in frequency response as well as semantic similarity in its reduction step. Singular Score can be computationally heavy as it calculates the SVD of the input embedding and will scale exponentially as the input dimension size increases.

5.3 Metrics

Intent Classification benefits from all data reduction techniques across different languages. This indicates that the model has abundance of training data for intent classification given the simpler nature of the task as compared to Slot Filling. In the joint IC/SF BERT model context, we see that intent classification accuracy improves while also showing improvements in SemER indicating no compromise on the Slot Filling task.

6 Conclusion

In this paper, we investigated various inexpensive approaches for identifying data redundancy in training data used to fine-tune BERT based models in SLU systems for the IC and SF task. To the best of our knowledge, this work is the first step in the direction of identifying inexpensive techniques to fine-tune BERT model without affecting offline metrics. We presented empirical results on a real-world SLU dataset, showing that data reduction techniques benefit SemER and Intent Classification metrics. In particular, we proposed a novel data redundancy identification and reduction technique which we call the Singular Score approach. This method helps jointly filter utterances based on their frequency and semantic representation and also helps achieve one of the best results among the techniques experimented with. Future work may target more complex forms of identifying training data redundancy such as influential instances (Pruthi et al., 2020; Koh and Liang, 2020) or active learning Grießhaber et al. (2020)

References

Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. 2021. Muppet: Massive multi-task representations with pre-finetuning.

Álvar Arnaiz-González, J. Díez-Pastor, Juan José Rodríguez Diez, and C. García-Osorio. 2016. Instance selection of linear complexity for big data. *Knowl. Based Syst.*, 107:83–95.

Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. 2019. Cloze-driven pretraining of self-attention networks.

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling.

Hyunjin Choi, Judong Kim, Seongho Joe, Seungjai Min, and Youngjune Gwon. 2021. Analyzing zero-shot cross-lingual transfer in supervised nlp tasks.

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.

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.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation.

Aysu Ezen-Can. 2020. A comparison of lstm and bert for small corpus.

George W. Furnas, Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, Richard A. Harshman, Lynn A. Streeter, and Karen E. Lochbaum. 1988. Information retrieval using a singular value decomposition model of latent semantic structure. In *SIGIR*, pages 465–480. ACM.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning.

Judith Gaspers, Penny Karanasou, and Rajen Chatterjee. 2018. Selecting machine-translated data for quick bootstrapping of a natural language understanding system. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 137–144, New Orleans - Louisiana. Association for Computational Linguistics.

Ruiying Geng, Binhua Li, Yongbin Li, Xiaodan Zhu, Ping Jian, and Jian Sun. 2019. Induction networks for few-shot text classification.

Gene H. Golub and Christian Reinsch. 1971. Singular value decomposition and least squares solutions. *Linear Algebra*, pages 134–151.

Daniel Grießhaber, Johannes Maucher, and Ngoc Thang Vu. 2020. Fine-tuning bert for low-resource natural language understanding via active learning.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification.

Jiaxin Huang, Chunyuan Li, Krishan Subudhi, Damien Jose, Shobana Balakrishnan, Weizhu Chen, Baolin Peng, Jianfeng Gao, and Jiawei Han. 2020. Few-shot named entity recognition: A comprehensive study.

W. Kabsch. 1978. A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 34(5):827–828.

Gregory R. Koch. 2015. Siamese neural networks for one-shot image recognition.

Pang Wei Koh and Percy Liang. 2020. Understanding black-box predictions via influence functions.

Andrew K. Lampinen and James L. McClelland. 2018. One-shot and few-shot learning of word embeddings.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual bert?

Garima Pruthi, Frederick Liu, Mukund Sundararajan, and Satyen Kale. 2020. Estimating training data influence by tracing gradient descent.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Sebastian Ruder, Matthew E. Peters, Swabha Swayamdipta, and Thomas Wolf. 2019. Transfer learning in natural language processing. In *NAACL-HLT*.

Eyal Shnarch, Ariel Gera, Alon Halfon, Lena Dankin, Leshem Choshen, Ranit Aharonov, and Noam Slonim. 2021. Cluster & tune: Enhance {bert} performance in low resource text classification.

Shyam Upadhyay, Manaal Faruqui, Gökhan Tür, Dilek Z. Hakkani-Tür, and Larry Heck. 2018. (almost) zero-shot cross-lingual spoken language understanding. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6034–6038.

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

S. Walton, O. Hassan, and K. Morgan. 2013. Reduced order modelling for unsteady fluid flow using proper orthogonal decomposition and radial basis functions. *Applied Mathematical Modelling*, 37(20-21):8930–8945.

D. Wilson and T. Martinez. 2004. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38:257–286.

Weijia Xu, Batool Haider, and Saab Mansour. 2020. End-to-end slot alignment and recognition for cross-lingual nlu.

Yi Yang and Arzoo Katiyar. 2020. Simple and effective few-shot named entity recognition with structured nearest neighbor learning.

Wenpeng Yin. 2020. Meta-learning for few-shot natural language processing: A survey.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q. Weinberger, and Yoav Artzi. 2021. Revisiting few-sample bert fine-tuning.

Leveraging Expectation Maximization for Identifying Claims in Low Resource Indian Languages

Rudra Dhar
Jadavpur University
West Bengal, India
rudradharrd@gmail.com

Dipankar Das
Jadavpur University
West Bengal, India
dipankar.dipnil2005@gmail.com

Abstract

Identification of the checkable claims is one of the important prior tasks while dealing with an infinite amount of data streaming from social web and the task becomes a compulsory one when we analyze them on behalf of a multilingual country like India that contains more than 1 billion people. In the present work, we describe our system which is made for detecting check-worthy claim sentences in resource scarce Indian languages (e.g., Bengali and Hindi). Firstly, we collected sentences from various sources in Bengali and Hindi and vectorized them with several NLP features. We labeled a small portion of them for check-worthy claims manually. However, in order to label the rest of data in a semi-supervised fashion, we employed the Expectation Maximization (EM) algorithm tuned with the Multivariate Gaussian Mixture Model (GMM) to assign weak labels. The optimal number of Gaussians in this algorithm is traced by using Logistic Regression. Furthermore, we used different ratios of manually labeled data and weakly labeled data to train our various machine learning models. We tabulated and plotted the performances of the models along with the stepwise decrement in proportion of manually labeled data. The experimental results were at par with our theoretical understanding, and we conclude that the weak labeling of check-worthy claim sentences in low resource languages with the EM algorithm has true potential.

1 Introduction

Misinformation has taken over the internet and it is now a well-recognized problem all over the world. Governments, news agencies, information security people all are trying to fight this menace with the helping hands from the researchers of social networks, natural language processing, data science and many more. With the exponential rise in information or news, making automated systems of fact-checking becomes a necessity. Researchers have made significant achievements in this field. But most of the research is in high resource languages like English. However, misinformation or fake news doesn't stop in the high resource language. Fake news is spreading far and wide in low resource languages as well, like in the Indian languages of Bengali and Hindi (Majumder and Das, 2020). A lot more research needs to be done in this arena and our present work we have contributed towards it.

It is a common practice in machine learning to use a semi supervised approach to weakly label data when labeled data is scarce (Xuan et al., 2010). On the other hand, filtering out check worthy claim sentences has been attempted by many teams as mentioned in (Hassan et al., 2015), (Dhar et al., 2019). While (Hassan et al., 2017) has made a generic and large system, (Anand et al., 2018) worked on twitter feeds. However, the problem of identifying claim sentences in Indian languages has not been attempted much and thus doesn't have a lot of labeled data. So here, we use the Expectation Maximization (EM) (Dellaert, 2003), a semi supervised algorithm to assign weakly labels to a lot of unlabeled data in Indian languages.

Suggested by (Hassan et al., 2015), automated fact checking is a two-fold task: 1) identification of check-worthy sentences, and 2) checking their trustworthiness based on some reliable sources. As checking the trustworthiness or veracity is a very resource intensive and computationally expensive job, identification or filtering out the check-worthy sentences becomes very important. In this work, we have used the semi supervised Expectation Maximization (EM) (Dellaert, 2003) algorithm to weakly label check-worthy sentences in the low resources Indian languages, Bengali and Hindi. Using semi supervised algorithms to label a lot of

	# web pages crawled	# sentence (labeled)	# sentence (unlabeled)	Average length of sentence in Words
Bengali	400	1500	8568	11.40
Hindi	270	902	13537	13.70

Table 1: Statistics of our data.

unlabeled data is a common practice in Machine Learning and NLP. But, using EM or similar algorithms to weakly label check-worthy sentences has rarely been tried, and has certainly never been attempted in the context of Indian languages.

The rest of the Sections are organized as follows. Section 2 gives the approaches in developing the datasets while Section 3 and Section 4 describe the algorithms of Expectation Maximization (EM) and GMM, respectively. The experimental results are shown in Section 5 along with important observations. Finally, Section 6 concludes the draft.

2 Data Set Preparation

We prepared our own data for this work. We crawled sentences (mostly news) from various web sources and manually labeled some of them. We crawled Bengali sentences from 'ABP Ananda' (https://bengali.abplive.com/) and Hindi sentences from 'Abp News' (https://www.abplive.com/) and 'Aajtak' (https://aajtak.intoday.in/), as well as 'Twitter' (https://twitter.com/?lang=bn/hn).

We manually labeled a portion of the data (about 1500 for Bengali, and about 900 for Hindi). There were two classes. Class 1 refers to check-worthy claims, whereas class 0 refers to not-check-worthy sentences(that is, the rest of the sentences). Only the sentences, which is a fact checkable claim, with a clear context, are put in class 1. For example the sentence ('It is to be mentioned that in 2014, Rahul defeated Smriti Irani by 1 lakh 6 thousand votes in Amethi constituency.') is considered a check-worthy claim and hence would be labeled as class 1. Whereas the sentence ('The court summoned him on Thursday for this issue') is not considered a check-worthy claim and hence would be labeled as class 0.

We extracted tf-idf scores, unigram, bi-gram and part of speech tags, as features for both the languages. It has to be mentioned that the fitting of data in a Multivariate Gaussian Mixture Model is a very computationally expensive job. Experimentation by using all the thousands of features in our Gaussian Mixture Model (19907 for Bengali, and 27928 for Hindi) was not possible in our available hardware setup. Therefore, we used Principal Component Analysis (PCA) to reduce the dimensionality of the data to 1000. For Bengali we retained 92.9% data whereas in Hindi we retained 96.6% data.

3 EM - Semi-Supervised Algorithm

It is described in (Dellaert, 2003) that EM is an iterative optimization method to estimate some unknown parameters Θ, given measurement data U. However, we are not given some "hidden" nuisance variables J, which needs to be integrated out. In particular, we want to maximize the posterior probability of the parameters Θ given the data U, marginalizing over J:

$$\Theta^* = \arg\max_{\Theta} \sum_{J \in \mathcal{T}^n} P(\Theta, J|U)$$

(Dempster et al., 1977) describes it from a statistical point of view in detail and (Little and Rubin, 2014) states its proof. The EM algorithm seeks to find the Maximum Likely Estimation of the marginal likelihood by iteratively applying the two steps namely **Expectation** and **Maximization**.

The EM algorithm needs a mixture model to represent the data. Here, we use 'Multivariate Gaussian Mixture Models' to represent the data. A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. 'Multivariate Gaussian Mixture Models' are Gaussian Mixture Models in a multivariable or multidimensional vector space.

4 Experimental Setup

Firstly, we separated out the validation and the test sets from the manually labeled data with a train-validation-test split of 60%-20%-20% in ratio. Then, we made the training set using (strongly/human) labeled training data plus some of the unlabeled data while the validation and the test set consist of entirely labeled data. In each step of Expectation Maximization, we fit the training

No. of Gaussian	**Class**	**Results on validation data**					
		Bengali			**Hindi**		
		precision	recall	F1	precision	recall	F1
8	0	0.78	0.53	0.63	0.75	0.56	0.64
	1	0.38	0.66	0.48	0.40	0.60	0.48
16	0	**0.72**	**0.59**	**0.65**	0.75	0.53	0.62
	1	**0.41**	**0.56**	**0.47**	0.38	0.61	0.47
32	0	0.78	0.53	0.63	**0.78**	**0.65**	**0.71**
	1	0.38	0.66	0.48	**0.48**	**0.62**	**0.54**
64	0	0.78	0.53	0.63	0.81	0.53	0.64
	1	0.38	0.66	0.48	0.42	0.74	0.54

Table 2: Results on validation data given different number of Gaussian

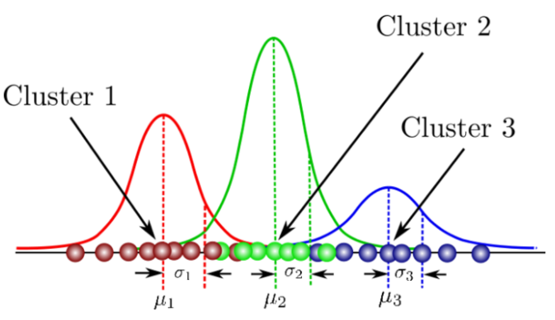


Figure 1: Single variable Gaussian mixture model with 3 Gaussian.

set to a Gaussian Mixture Model with a chosen number of Gaussians and assign the weak labels to the unlabeled data. Then, we measured the efficiency of the Gaussian Mixture Model by training a logistic regression component and evaluated on the validation set. We repeat this step and select the GMM which gives the best results on the validation set. At last, we report the performance of the selected GMM on the test set by modeling various Machine Learning models using human labeled data plus weakly labeled data. In addition, we have also explored this for various proportions of human (strongly) and weakly labeled data, and observed the performances with respect to the change in proportion. The details of each step are given in the following subsections as pseudo-code. (Note that we denote check-worthy claim sentences as class '1' and rest sentences as class '0') (shown in Figure 2).

4.1 Pseudo-code of the EM algorithm

```
• X_train = X_train_labeled + X_train_unlabled
• threshold = (number of claim in labeled data)
  / (number of labeled data)
  #threshold is the proportion of claims in labeled data. Later we will mark a set to be claim set, if it has higher proportion of claims than threshold
• choose 'gaussianNumber' as the number of Gaussians in our GMM.
• gModel = fit GMM with X_train
• get which training example is assigned to which Gaussian:
  yGaussian = gModel .predict(X_train)
• from yGaussian calculate number of strongly labeled '0' and '1' that fell under each Gaussian:
  for each Gaussian:
    – gy[0] = number of strongly labeled '0' in the Gaussian
    – gy[1] = number of strongly labeled '1' in the Gaussian
• calculate proportion of strongly labeled '1' in a Gaussian from gy
  for each Gaussian:
    – gRatio = gy[1] / (gy[0] + gy[1])
• put a label on each Gaussian:
  for each Gaussian:
    – if gRatio > threshold, then gaussianLabel = 1
    – else gaussianLabel = 0
• weakly label unlabeled data:
  y_train_weak_labled = 'gaussianLabel' of the Gaussian assigned to the sample.
```

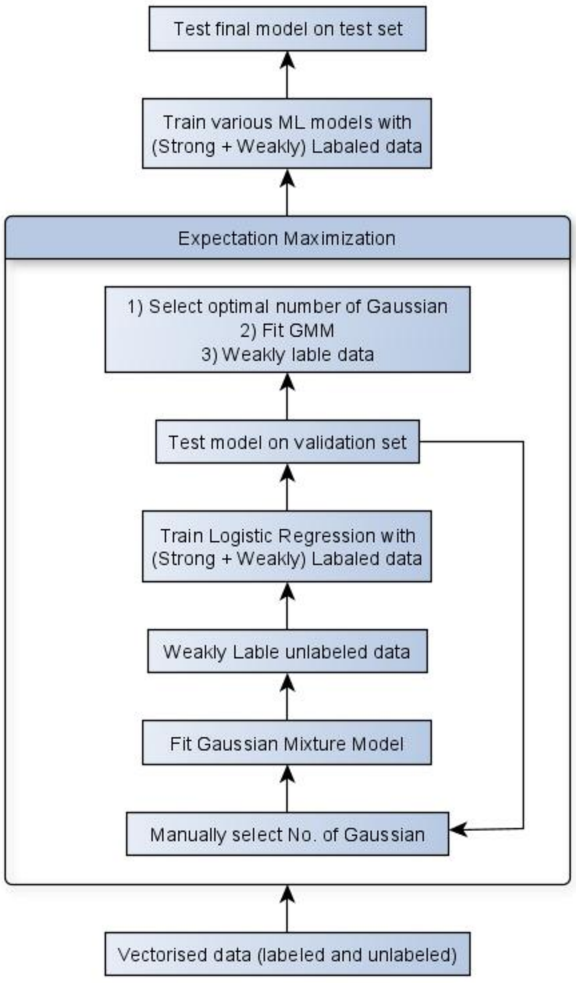

Test final model on test set
Train various ML models with (Strong + Weakly) Labaled data
Expectation Maximization
1) Select optimal number of Gaussian
2) Fit GMM
3) Weakly lable data
Test model on validation set
Train Logistic Regression with (Strong + Weakly) Labaled data
Weakly Lable unlabeled data
Fit Gaussian Mixture Model
Manually select No. of Gaussian
Vectorised data (labeled and unlabeled)

Figure 2: Architecture of the system including the EM algorithm.

- y_train = y_train_labled + y_train_weak_labled
- LModel = Logistic Regression model trained by(X_train, y_train)
- predicted_validation = predict value on validation set by LModel
- classification_report(true_validation, predicted_validation)

5 Experiments and Evaluation

We considered various Gaussian Mixture Models, with different numbers of Gaussian and selected the model which has the best performance on the validation data. We enlist the precision, recall, and the F1 score of the models on the validation data for both the classes and both the languages in Table 2. We manually observed these results and tried to determine the optimum number of Gaussians. We have noticed that the Gaussian mixture model with 16 numbers of Gaussians has the most promising results for Bengali while for Hindi a model with 32 numbers of Gaussian achieves the best result.

5.1 Training and Testing

After fixing the optimal Gaussian mixture model with the help of logistic regression, we train various machine learning models with human labeled data merged with semi-supervised weakly labeled data. However, we have fixed the number of human labeled data in the training set throughout the experiments. While we take various amounts of weakly labeled data and add them into the training set for identifying the impacts on utilizing variations in size of the weakly labeled data. We trained a logistic regression as well as Support Vector Machine (SVM) classifiers which are the classic machine learning models along with a neural network (Multi Layer Perceptron), and compared the results. For the neural networks, we used various MLP and chose the one giving the best results. In Table 3.1 and 3.2, we noted the performances of various classifiers on the validation set vs. size of weakly labeled data in the training set. (Note that class 1 is check-worthy claims, and class 0 is not-check-worthy claims) The number of weakly labeled data vs. the F1 score obtained in the test set is plotted in Figure 3, for both the languages and both the classes.

5.2 Observations

We observed that the performances of the models decrease with respect to the reduction in the proportion of human annotated data in the training set. This is also very much expected, as any machine learning algorithm's performance is supposed to deteriorate as the quality of its training data deteriorates.

This decrease in performance with increase in proportion of weakly labeled data is steeper in case of Hindi while for Bengali, it is more gentle. The gentle decline of performance with the increase in proportion of weakly labeled data for the Bengali dataset is an important observation. It shows that, annotating a lot of data with weak labels by employing a semi supervised algorithm, for detecting fact checkable claims is possible.

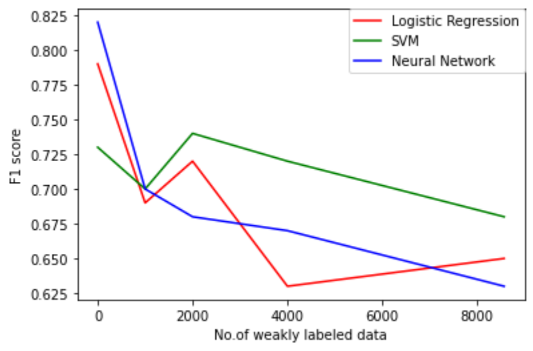
In the case of Bengali for both the classes, the performances of the classifiers like LR and SVM started improving while adding a weakly labeled data of size 1K to 2K while the neural networks

Size of Weakly Labeled Data	class	LR	SVM	Neural network	No. of sample in support
0	0 1	0.79 0.54	0.73 0.49	0.82 0.54	210 90
1000	0 1	0.69 0.45	0.70 0.40	0.70 0.49	210 90
2000	0 1	0.72 0.46	0.74 0.44	0.68 0.47	210 90
4000	0 1	0.63 0.38	0.72 0.44	0.67 0.46	210 90
8568	0 1	0.65 0.43	0.68 0.44	0.63 0.46	210 90

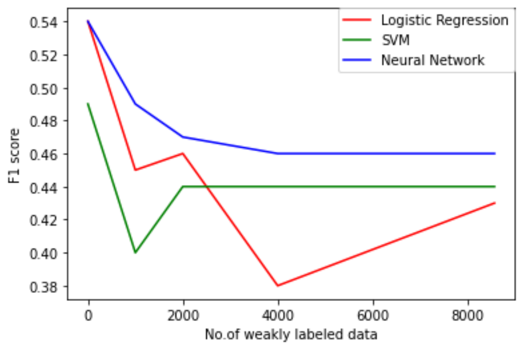
Table 3: F1 score for different classifiers with respect to different amounts of weakly labeled data in **Bengali**.

Size of Weakly Labeled Data	class	LR	SVM	Neural network	No. of sample in support
0	0 1	0.77 0.60	0.73 0.56	0.84 0.62	123 58
1000	0 1	0.82 0.61	0.74 0.55	0.83 0.68	123 58
2000	0 1	0.77 0.63	0.74 0.51	0.76 0.61	123 58
4000	0 1	0.71 0.60	0.71 0.47	0.73 0.54	123 58

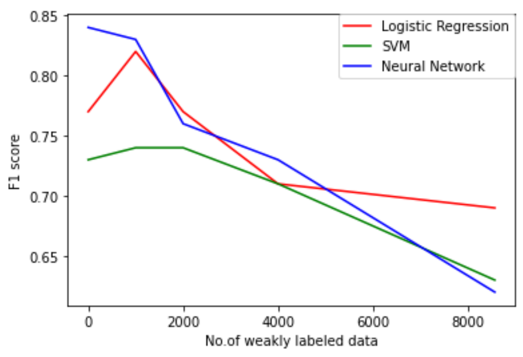
Table 4: F1 score for different classifiers with respect to different amounts of weakly labeled data in **Hindi**.



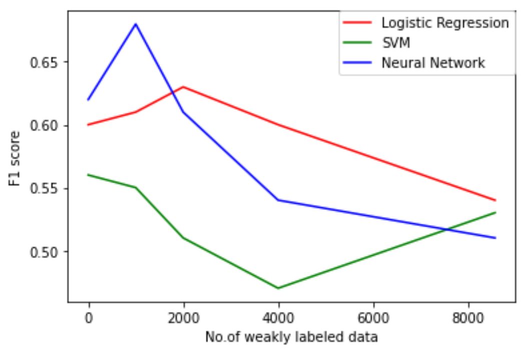
(a) Bengali class 0



(b) Bengali class 1



(c) Hindi class 0



(d) Hindi class 1

Figure 3: F1 score on test set vs No. of weakly labeled data in training set.

started degrading. In contrast, LR and neural networks started improving in case of Hindi data in the range of 500 to 1K except SVM. This is probably due to the fact that most machine learning algorithms perform better with more data. Thus, when a little amount of weakly labeled data is added to the training set, the performance of some of the models are improved. But, the performances of some of the models do not improve owing to the fact that the data we are adding is ultimately weakly labeled, and not by a human annotator.

6 Conclusion

In the present work, we have attempted to filter out check-worthy claims, which is the first step of fact-checking. Since there is a lot of information, and little intelligent labour available for manually tagging check-worthy sentences, we have used a semi-supervised algorithm to quickly tag a lot of check-worthy sentences by achieving a satisfied level of accuracy. We used the semi supervised Expectation Maximization algorithm with Gaussian Mixture Models, to weakly label the unlabeled data. We crawled a lot of data from the web and weakly labelled them using the Expectation Maximization algorithm. We trained some classification models with this weakly labeled data and observed the results on human annotated data. We find that the performance is very close to the performance by training with manually labeled data. Therefore, we can conclude that our semi supervised algorithm works well in the task of identifying fact checkable sentences.

We expect the efficiency will increase in future with more complicated models and more labeled data. For the Expectation maximization algorithm, computation power is also a concern. It is difficult to fit Gaussian Mixture models with high dimensions with currently available hardware. However we did proof that this approach works for identifying a lot of check-worthy claims with fair accuracy in Indian languages.

Acknowledgments

The present work is supported by the research project entitled ”Claim Detection and Verification using Deep NLP: an Indian Perspective” funded by DRDO, Government of India.

References

Sarthak Anand, Rajat Gupta, Rajiv Ratn Shah, and Ponnurangam Kumaraguru. 2018. Fully automatic approach to identify factual or fact-checkable tweets. In *FIRE*.

Frank Dellaert. 2003. The expectation maximization algorithm.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the em - algorithm plus discussions on the paper.

Rudra Dhar, Subhabrata Dutta, and Dipankar Das. 2019. A hybrid model to rank sentences for check-worthiness. In *CLEF*.

Naeemul Hassan, Bill Adair, James Hamilton, Chengkai Li, Mark Tremayne, Jun Yang, and Cong Yu. 2015. The quest to automate fact-checking. *Proceedings of the 2015 Computation + Journalism Symposium*.

Naeemul Hassan, Anil Nayak, Vikas Sable, Chengkai Li, Mark Tremayne, Gensheng Zhang, Fatma Arslan, Josue Caraballo, Damian Jimenez, Siddhant Gawsane, Shohedul Hasan, Minumol Joseph, and Aaditya Kulkarni. 2017. Claimbuster: the first-ever end-to-end fact-checking system. *Proceedings of the VLDB Endowment*, 10:1945–1948.

Roderick Little and Donald Rubin. 2014. *Statistical Analysis with Missing Data, Second Edition*, pages 200–220.

Soumayan Bandhu Majumder and Dipankar Das. 2020. Detecting fake news spreaders on twitter using universal sentence encoder. In *CLEF*.

Jifeng Xuan, Jiang He, Zhilei Ren, Jun Yan, and Zhongxuan Luo. 2010. Automatic bug triage using semi-supervised text classification. pages 209–214.

Performance of BERT on Persuasion for Good

Saumajit Saha, Kanika Kalra, Manasi Patwardhan, Shirish Karande
TCS Research Pune, India
{saha.saumajit, kalra.kanika, manasi.patwardhan, shirish.karande}@tcs.com

Abstract

We consider the task of automatically classifying the persuasion strategy employed by an utterance in a dialog. We base our work on the PERSUASION-FOR-GOOD dataset, which is composed of conversations between crowdworkers trying to convince each other to make donations to a charity. Currently, the best known performance on this dataset, for classification of persuader's strategy, is not derived by employing pretrained language models like BERT. We observe that a straightforward fine-tuning of BERT does not provide significant performance gain. Nevertheless, non-uniformly sampling to account for the class imbalance and a cost function enforcing a hierarchical probabilistic structure on the classes provides an absolute improvement of 10.79% F1 over the previously reported results. On the same dataset, we replicate the framework for classifying the persuadee's response.

1 Introduction

With the advancement of artificial intelligence, there has been a tremendous rise in its usage in the daily lives of people. Birth of conversational agents has made life a lot easier for organizations looking to achieve certain tasks. These agents, like as mentioned in (Luger and Sellen, 2016; Bickmore et al., 2016; Graesser et al., 2014), are goal-oriented, *i.e.*, they try to engage users in meaningful conversations and thereby aim to achieve their tasks. At times, they require different strategies of persuasion in order to mould people into their way of thinking, thereby changing their specific attitude or behaviour (Shi et al., 2020). Wang et al. (2019) proposed the foundation on building an automatic personalized persuasive dialogue system. They created the PERSUASION-FOR-GOOD dataset, which is composed of conversations between crowdworkers trying to convince each other to make donations to a charity. They annotated the utterances with persuasive strategy labels and proposed a baseline method for persuasive strategy classification.

Until recently, the dominant prototype in approaching any natural language processing tasks has been to focus on designing neural network architectures, using task specific data and word embeddings such as GloVe (Pennington et al., 2014). The NLP community is witnessing a paradigm shift towards pre-trained deep language representation model which achieves the SOTA in question answering, sentiment analysis and other NLP tasks. Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) represents one of the latest developments in this field. It surpasses its predecessors, ELMo (Peters et al., 2018) and GPT (Radford et al.) by a significant margin on numerous NLP tasks. There is not much literature on exploring BERT for tasks related to persuasive dialogues.

In this work, we introduce a BERT-based approach to automatically classify the persuasion strategy employed by an utterance in a dialog. We also use the same approach to classify the type of the response of the persuadee's utterance. We base our work on the PERSUASION-FOR-GOOD dataset. Since the amount of annotated dialogues in this dataset are very less, we experiment to evaluate the efficacy of pretrained BERT in achieving better performance for the said task. The main contribution of this work is : 1. Creating a BERT-based hierarchical classification setup for classification of Persuader's strategy . 2. Creating a benchmark setup for the Persuadee's response classification 3. Additional analysis for the dataset introduced by (Wang et al., 2019).

The baseline performance for strategy classification on PERSUASION-FOR-GOOD is an F1 score of 59.6% and 74.8% accuracy (Wang et al., 2019). We observe that a straightforward fine-tuning of BERT does not provide significant performance gain: 60.60% F1 and 75.85% accuracy. Nevertheless, non-uniformly sampling to account for the class imbalance does improve performance to 68.1% F1 and 77.69% accuracy. We further employed a cost function which enforces a hierarchical probabilistic structure on the classes, namely a utterance is Persuasive or Not, and if persuasive, the strategies further belong to coarser classes of Appeal or Inquiry. This step improves the performance to 70.39% F1 and 79.50% accuracy, which is an absolute improvement of 10.79 F1 over previously reported results.

The remainder of this paper is organized as: We review the related work in Section 2. In Section 3 we conduct data analysis. In Section 4 we describe our methodology. We analyze the results and present our observations in Section 5. Finally, we summarize the key conclusion in Section 6.

2 Related Work

2.1 Persuasive Conversation

Several previous works have looked at detecting persuasion in online forums and social networks. (Yu et al., 2019) performed fine-grained analysis of texts by detecting all fragments that contain propaganda techniques as well as their type in news articles. (Morio et al., 2019), (Tan et al., 2016) (Hidey and McKeown, 2018) worked on persuasion detection on online forum by modeling argument sequence in social media. (Yang et al., 2019) focuses on persuasive strategy detection in semi supervised fashion for sentences of posts on a crowd funding platform. Meanwhile, the number of papers which have attempted mining persuasive strategies in dialog conversations have been limited. (Keizer et al., 2017) evaluated persuasion as a strategy for negotiating dialog agent. However, we believe that the recent work by Wang et al. (2019) is a first attempt to explicitly collect corpus of persuasive conversations. They collect a persuasive conversation dataset(PERSUASION-FOR-GOOD) for charity donation with persuasive strategy annotations. Our work in this paper is based on this dataset. In recent years there has also been interest in generating persuasive utterances and slogans. (Munigala et al., 2018) generate persuasive captions for fashion items on an e-commerce website. (Li et al., 2019) generate dialogs based on the PERSUASION-FOR-GOOD dataset. Meanwhile, (Shi et al., 2020) have developed a retrieval based persuasive dialog agent with the same dataset. The scope of this work is however limited to strategy classification.

2.2 Hierarchical Classification

There are multiple ways the literature has exploited the hierarchical structure of the labels to improve the performance. (Kowsari et al., 2017) uses local models, viz. one for each node in the label hierarchy, where the lower level classifiers are stacked on top of the higher level. The inference is made using top-down strategy. There are flat approaches (Charuvaka and Rangwala, 2015; Xu and Geng, 2019), which employ one model per leaf node. They perform cost-sensitive classification by penalizing the mis-classification of negative examples as per their distance from the training class in the hierarchy. These approaches require multiple models for classification. Hence, using these approaches with the BERT based classification technique we have employed would be resource intensive.

There are techniques which take label embeddings into consideration. For example, (Rios and Kavuluru, 2018; Pal et al., 2020) approach of document classification, uses variants of Graph Neural Networks to embed the hierarchical information of the label space and employs information retrieval setting to match these label embeddings with the document vectors. In our method, Multilabel Classification with Probabilistic Structure (MLPS) described in section 4.2, we enforce the hierarchical probabilistic structure on the class predictions rather than the label embeddings.

Some local approaches (Gopal and Yang, 2013; Peng et al., 2018) employ regularization technique by constraining the parameters of the parent and child classifiers to be similar. Instead of using distinct set of parameters for the parent and the child, Banerjee et al. (2019) introduce inductive bias by initializing the parameters of the finer level classifiers with the parameters of the coarser level classifier and further fine-tuning them on the finer category classification. Our current baseline method for multil-label classification (ML), as described in section 4.2, does not take the advantage of the label hierarchies. In future we would like to extend this method to be on the lines of (Molino et al., 2018; Patidar et al., 2018), which model the label dependencies by using a sequential decoder to predict a branch of the labels in the hierarchy.

3 Dataset

Wang et al. (2019) designed an online task to col-

Binary Coarser Label	Ternary Coarser Label	Strategy Label
Persuasive	Persuasive Appeal	1) Emotional-appeal 2) Foot-in-the-door 3) Logical-appeal 4) Personal-story 5) Credibility-appeal 6) Donation-information 7) Self-modeling
	Persuasive Inquiry	8) Task-related-inquiry 9) Personal-related-inquiry 10) Source-related-inquiry
Non Persuasive	Non Persuasive	11) Non-strategy

Table 1: Persuader Strategy Label Hierarchy.

Coarser Label	Actual Label
Ask	1) Ask-org-info, 2) Ask-donation-procedure
Positive	3) Positive-to-inquiry, 4) Positive-reaction-to-donation, 5) Agree-donation
Negative	6) Disagree Donation, 7) Disagree Donation more, 8) Negative reaction to donation, 9) Negative to inquiry
Neutral	10) Neutral to inquiry, 11) Neutral reaction to donation
Greeting	12) Greeting
Other	13) Other, 14) Off-task, 15) Acknowledgement, 16) closing, 17) you-are-welcome, 18) thank
Task-related-inquiry	19) Provide donation amount, 20) confirm donation, 21) task-related-inquiry
Personal-related-inquiry	22) Ask persuader donation intention, 23) personal-related-inquiry

Table 2: Persuadee Response Label Hierarchy.

lect the persuasive dialog data. The main objective of the task was to persuade the other person to donate some amount for the charity *Save the Children*[1]. This task was performed by two participants, where one person who tries to persuade the other person for donation is termed as the persuader and the other person who donates is referred to as the persuadee. The persuader was instructed to use different types of strategies for persuading the persuadee. After the dialogue is over, both persuader and persuadee can choose to donate amount for charity. The chosen amount gets deducted from their task payment on Mturk.

This data collection process involved 1285 participants acting as either persuader or persuadee. After collecting the data, the authors annotated 300 dialogues out of 1017 with labels of persuasive strategy for each of the persuader's utterances in a dialogue. The annotated dataset consists of 10 persuasive strategies and one non-strategy class corresponding to persuader's utterances. The persuasive strategies listed in Table 1 were broadly categorized as persuasive appeal and persuasive inquiry. Each response of a persuadee was also annotated into one of the 23 different classes listed in Table 2.

The 300 dialogues annotated in Wang et al. (2019) are used to setup a persuasion strategy classification task. Each sample consists of the current persuader utterance and prior persuadee utterance which is considered as context. The sample has been labeled with one of the 11 strategy classes. We use these labels to associate coarser labels with each samples in accordance to Table 1. For e.g,

Context : That's so important. How do you raise donations?

Input : Do you currently donate to your charity?

Strategy label : Task-related inquiry

Coarser labels : Persuasive, Persuasive Inquiry

Wang et al. (2019) have provided a 5-fold data-split

[1]https://www.savethechildren.org/

such that the training set contains 3450 utterances and the validation set contains 863 utterances. The results presented in this paper, unless specified otherwise are based on these exact data splits.

We also address another task of Persuadee's response classification. We have created 5-fold splits from 300 annotated dialogues for persuadee response classification task in the manner similar as for Persuader's strategy classification. This contains 3877 samples in the training set and 970 samples in the validation set. Following is an example for such a task:

Context : Would $2.00 be too much to ask?

Input : No, I can do it.

Response type : Agree-donation

Coarser Label : Positive

3.1 Dataset Analysis

3.1.1 Dialog Independence

Wang et al. (2019) conducted a survey to categorize the personalities of the crowd workers, and did not notice a significant correlation with the choice of strategy. Nevertheless, if a human participates in several conversations then one could learn identity specific preferences for language usage and dialog strategy. We found out that 524 participants acted as only persuader, 584 acted as only persuadee and 177 participants acted as both. Figure 1 shows the count of conversation a participant participated in based on the acted role. This depicts that there are few participants which took part in more than one conversation. Hence, even though we believe that the modeling of identities can help personalize persuasion understanding, in this work, we do not take identities as input to our models, and train a single model which works only on utterances.

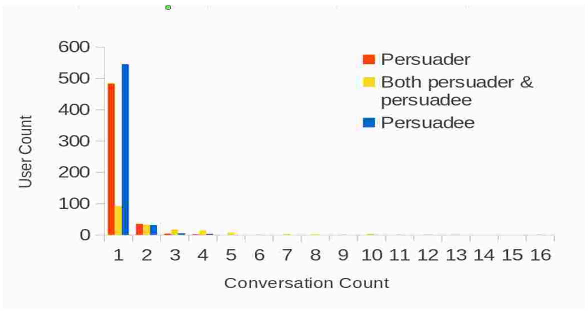


Figure 1: User role wise conversation participation count.

3.1.2 Interdependence of Responses

We also investigated for dependency among labels for both persuader and persuadee as well as current utterance and prior utterance labels individually for persuader and persuadee. We merged the persuadee class labels together to form 8 coarser class labels in the similar fashion as it was done for persuader strategies. Table 2 depicts the coarser label and the corresponding actual labels. We found out the sample mutual information by considering 8 persuadee class labels and 11 persuader strategy labels. The mutual information between persuader's current and prior utterance label is 0.1091 which indicates that persuader's current strategy is not influenced by prior strategy. The mutual information between persuadee's current and prior utterance label is 0.1222 which indicates that persuadee's current utterance response is not influenced by prior response. The mutual information between the persuader's current utterance label and persuadee's prior utterance label is 0.1452 which also indicates that persuader's strategy is not highly influenced by persuadee's response. These observations are reflected in our results section, where we did not observe significant advantages by including dialog history for classifying a particular utterance.

3.1.3 Truthfulness of Dialogues

We found that out of 643 persuadees participating in single conversation; only 355 donated and 288 did not donate. Further, out of the remaining 118 persuadees participating in more than one conversation; 41 donated in each of the conversation, 46 did not donate at all and rest donated in some of the conversations. Similar analysis was done for persuader's role as persuader can also agree to donate in conversation in order to persuade persuadee. We have found that out of 575 persuaders participating in single conversation; only 242 donated and 333 did not donate. Further, out of the remaining 126 persuaders participating in more than one conversation; only 39 donated in each of the conversation and 66 did not donate at all and rest donated in some of the conversations. In a dataset such as this, a particular conversation should ideally be considered persuasive in nature if and only if the persuadee donated amount for charity because of participating in the dialogue with persuader. Such causal analysis may prove particularly challenging as based on manual inspection of the data we noted that few workers did not make donations despite agreeing to do so in the conversation.

4 Methodology

4.1 BERT

The model architecture of BERT is a multilayer bidirectional Transformer encoder based on the original Transformer model (Vaswani et al., 2017). The input representation of the BERT can distinctly represent a pair of sentences as a sequence of tokens. For each token, its input representation is constructed by summing the wordpiece embedding (Wu et al., 2016), segment and the position embeddings. Segment embeddings help to distinguish one sentence from the other in the pair. A special classification [CLS] token is inserted in the beginning of the sequence. A separator [SEP] token is inserted at the end of each sentence in the pair. Finally, the final hidden state representation of the [CLS] token of each sequence can be used for the sentence classification tasks.

4.2 Multilabel Classification with Probabilistic Structure

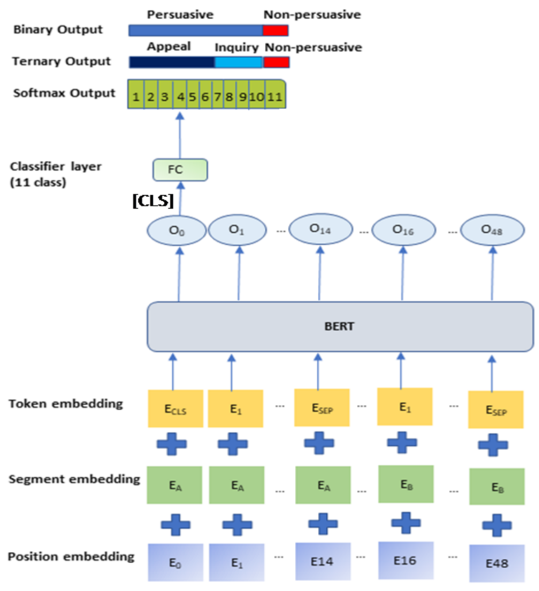


Figure 2: MLPS approach.

In persuader strategy classification task, input to BERT consists of the prior persuadee utterance as context and the current persuader's utterance. As shown in Figure 2 we use the final hidden state representation corresponding to the special [CLS] token as the aggregate representation of the input and pass it to a linear layer with *softmax* as its activation function. Finally, the posterior probability of each strategy is estimated by the softmax function P = **softmax**(WZ^T) where W is the weight matrix, $W \in \mathcal{R}^{d \times k}$ where k is the total number of strategies, d is the dimension of the [CLS] representation and Z is the representation of the final hidden state of the [CLS] token.

Softmax output is a k-dimensional vector, from which we choose the strategy corresponding to the highest value as our desired output. The first ten elements of this vector correspond to the persuasion strategies while the eleventh is the estimate for the probability that the utterance does not contain any persuasion. Furthermore, the first seven elements correspond to persuasive appeals. The posterior probabilities for the coarser labels can thus be estimated as:

$$P(Appeal) = \Sigma_{k=1:7} P_{\text{k}} \tag{1}$$

$$P(Inquiry) = \Sigma_{k=8:10} P_{\text{k}} \tag{2}$$

$$P(Persuasive) = \Sigma_{k=1:10} P_{\text{k}} \tag{3}$$

During inference, at each label granularity, the label with largest estimate for the posterior probability is chosen. We highlight that one can train only on the 11 fine granular strategy classes, and yet conduct inference for all the coarser labels. We consider three label sets for training (a) $\lambda_{(11,\_,\_)}$ where only the 11 fine-granular strategy labels are used for training (b) $\lambda_{(11,\_,2)}$ where we additionally use the coarser binary label persuasive or not (c) and finally $\lambda_{(11,3,2)}$ where we further utilize the ternary labels (appeal, inquiry or non-persuasive) for training. The total loss is a weighted sum of the cross-entropy loss over labels at each granularity. Without loss of generality we enforce that the weights sum to one, and perform a grid search to identify the best performing weight combinations for above approaches (b) and (c). We restrict our search to the part of the grid where the weight for the binary classification is the highest. The intuition behind this is that if the network, once learns to correctly predict binary label as it is at coarser level, would further improve the multi-class fine granular prediction. In the remainder of the paper we refer to approaches (b) and (c) as multilabel classification with probabilistic structure (MLPS). A similar approach has been adopted for classification of Persuadee responses, and the output of softmax for 23 classes is binned together in accordance to Table 2. We consider an additional approach as shown in Figure 3 to create a baseline for multilabel classification, to understand the utility of specifically including hierarchy or label interdependency. In this baseline the output of the

[CLS] pin is given to label specific linear layers with dimension equal to number of classes associated with the label. In the remainder we refer to this approach as multilabel classification (ML).

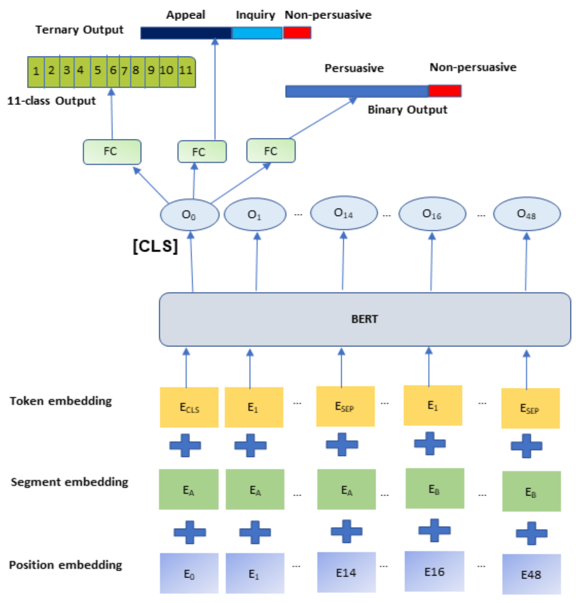


Figure 3: ML approach.

4.3 Turn Embedding

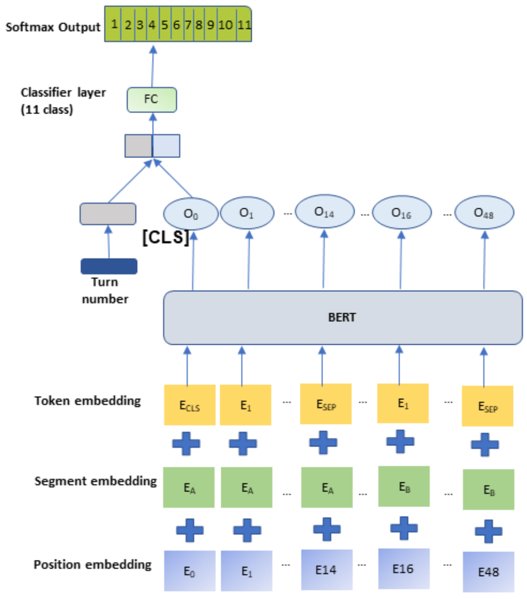


Figure 4: Architecture of Bert FT + Context + Turn

Wang et al. (2019) have shown that the distribution of employed strategies changes with the turn in a dialogue. We consider an ablation where we explore the utility of the turn side-information as an additional input to the persuader strategy classifier model. We consider a simple model where the 1-hot encoding for the turn is given to a hidden layer, which is concatenated to the [CLS] output of the final layer, before being fed to the linear layer before softmax. Turn embeddings did not prove beneficial when used in such a fashion, hence we did not consider them for multilabel classification. However, in future, it may be worth considering other approaches for embedding the turn information. Figure 4 refers to the approach which uses turn embedding.

5 Results and Analysis

5.1 Training Details

We use the pre-trained uncased BERT-base[2] model for fine-tuning. It consists of 12 Transformer blocks, its hidden layer size is 768, the number of self-attention heads present is 12, and the total number of parameters for the pretrained model is 110M. When fine-tuning, we keep the dropout rate to be 0.5, batch size to be 32 and the learning rate to be 2e -5. w_1, w_2 and w_3 are the weights of the loss functions, chosen in such a manner that they sum to 1. To tackle class imbalance, we have used Weighted Random Sampling (Efraimidis and Spirakis, 2008). We assign weights to the sampler such that each target label is assigned a weight equal to the reciprocal of the number of instances in the training set belonging to that target label. We have used Pytorch (Paszke et al., 2019) library while coding in python. Sequences of context and utterance having length greater than the maximum sequence length are truncated till 128. Shorter sequences are padded till the maximum sequence length. We use Early stopping in order to prevent over-fitting.

Our grid search revealed that: For persuader strategy classifier (a) for MLPS approach with training labels (11,_,2) best w_1 and w_2 are 0.4 and 0.6 respectively (b) for ML approach with training labels (11,_,2), w_1 and w_2 are 0.5 and 0.5 respectively (c) For MLPS approach with training labels (11,3,2), best result is found when w_1, w_2 and w_3 are 0.1, 0.3 and 0.6 respectively. (d) Similarly for ML approach with training labels (11,3,2), best result is obtained when w_1, w_2 and w_3 take the values 0.3, 0.3 and 0.4 respectively. For persuadee response classification (a) For MLPS approach, with training labels (23,8,_), best result is obtained when w_1 and w_2 are 0.1 and 0.9 respectively. (b) When w_1 and

[2]https://storage.googleapis.com/bert_models/2018_10_18/uncased_L-12_H-768_A-12.zip

Persuader's Strategy Classifier								
Method	**# Training Labels**	**Acc-11**	**F1-11**	**Acc-3**	**F1-3**	**Acc-2**	**F1-2**
Hybrid RCNN (Wang et al., 2019)		74.8%	59.6%				
Bert FT	11,_,_	74.65%	66.36%	89.36%	88.13%	89.59%	88.69%
Bert FT + Context	11,_,_	77.69%	68.1%	89.48%	88.78%	90.46%	89.94%
Bert FT + Context + ML	11,_,2	77.82%	68.69%	89.39%	88.5%	89.88%	89.23%
Bert FT + Context + ML	11,3,2	78.41%	69.30%	89.81%	88.71%	91.11%	90.6%
Bert FT + Context + MLPS	11,_,2	78.61%	68.99%	90.11%	89.11%	91.13%	90.73%
Bert FT + Context + MLPS	11,3,2	**79.50%**	**70.39%**	**90.59%**	**90.07%**	**91.49%**	**91.08%**
Persuadee's Response Classifier							
Method	**# Training Labels**	**Acc-23**	**F1-23**	**Acc-8**	**F1-8**		
Bert FT + Context	23,_,_	57.66%	46.05%	68.00%	59.48%		
Bert FT + Context + ML	23,8,_	56.60%	47.08%	68.71%	62.44%		
Bert FT + Context + MLPS	23,8,_	**61.80%**	**54.18%**	**71.5%**	**66.63%**		

Table 3: Persuader Strategy Classification and Persuadee Response Classification Results (Acc-N: Accuracy for N classes, F1-N: F1 Score for N classes).

w_2 both take the value of 0.5, we get the best result with ML approach.

5.2 Ablation Study

Experiments	**Acc**	**F1**
Without sampler	75.85%	60.60%
With sampler	77.69%	68.1%
Only Utterance and no Context	74.65%	66.36%
Without Turn	76.03%	67.23%
With Turn	75.51%	67.64%
Persuasive/Non-persuasive classification with Utterance and History	88.63%	88.20%

Table 4: Results of ablation experiments for BERT baseline for single label.

We have conducted various ablation studies with the baseline Bert based classifier for persuader strategy classification. Table 4 shows the results of this study. We have trained model with and without weighted random sampler. As the dataset is highly imbalanced, we observed that non-uniform sampling improves F1-score significantly. Thus, all the experiments reported in Table 3 are with sampling using (Efraimidis and Spirakis, 2008). We have also seen the importance of context alongside the current utterance as input. There has been an improvement of 3.04% in accuracy and 1.74% in the F1-score. We have also incorporated turn information as shown in Figure 4 and observed no significant improvement. Finally, we trained the model only for binary classes and observed that the result improves when trained jointly with multilabel as reported in Table 3.

5.3 Multilabel Strategy classification

Table 3 presents the results for our experiments on multilabel classification. We observe that with non-uniform class sampling the baseline training for BERT on the 11 persuasion strategy classes, provides a performance of 66.36% F1. This is an improvement of 6.76% over the previously reported result. We also observe that even if label inter-dependencies are used only at inference time for the coarser labels, one can still get a performance comparable to directly training just for that label. The performance of both multiobjective approaches, ML and MLPS, was observed to be better than the BERT baseline approach. Thus including additional structure during training helps performance for all labels. We further observe that MLPS provides a performance better than ML demonstrating the utility of including probabilistic structure in the cost function. The best performance for classification on 11 persuasion strategies was observed to be 70.39% F1, with MLPS and labels (11,3,2). This is an improvement of 10.79% over the previously reported result, of 4.03% F1 over the BERT baseline and of 1.09% over the multilabel baseline.

As illustrated in Table 5, we calculated the class-wise F1-scores for persuader strategy classification with the baseline BERT-FT model, as well as ML and MLPS with $\lambda_{(11,3,2)}$ label set. We made the

Class Label	BERT-FT	ML (11,3,2)	MLPS (11,3,2)
Personal story	27.58%	43.25%	40.99%
Logical-appeal	46.47%	58.31%	58.89%
Task. inquiry	51.35%	54.99%	51.57%
Self-modeling	56.17%	74.01%	70.03%
Personal. inquiry	65.51%	67.31%	70.53%
Foot-in-the-door	68.75%	68.6%	67.44%
Emotional-appeal	74.13%	76.38%	73.89%
Donation info.	76.43%	78.47%	77.74%
Credibility-appeal	81.67%	83.61%	85.2%
Non-strategy	86.81%	85.76%	88.31%
Source. inquiry	87.09%	86.32%	90.68%

Table 5: F1 score for each of the class label in Persuader's strategy classification

following observations: (i) % change in F1-score over the baseline is positive for most of the classes for both MLPS and ML indicating both the methods provide a performance gain across classes (ii) the % gain in F1-score for MLPS is more evenly distributed across all the classes as compared to the gain for ML, and (iii) the classes with lower baseline F1-scores are roughly getting benefited more in MLPS setting than the ones with higher baseline F1-score. These observations for MLPS are consistent with those made in (Banerjee et al., 2019; Peng et al., 2018), which say that exploiting the hierarchical structure benefits all fine granular classes.

5.4 Response classification

Wang et al. (2019) have not provided any baseline for automatic classification of Persuadee's responses. We observe that the use of BERT with weighted sampling for class imbalance provides an accuracy of 57.66 % and F1 score of 46.05% over the 23 response classes. MLPS provides an improvement of 4.14 % in accuracy and 8.13 % in F1. MLPS also improves over the bases for the 8 coarser classes. The improvement in accuracy is 3.5 % and 7.15 % in F1. ML provides an improvement of 1.03% in F1 while the accuracy has slightly decreased for 23 classes. However for the coarser classes, accuracy has increased by 0.71% and F1 has improved by 2.96%. On the similar lines of persuader strategy classification, we calculated the class-wise F1-scores for persuadee response classification. We observed that the % gain in the F1-score is more evenly distributed (i.e. lower standard deviation) across all the labels for MLPS when compared to ML.

5.5 Comparative Analysis with Examples

There have been several instances where one approach outperforms another approach as shown in Table 5. This section provides some of the examples, in Persuader's strategy classification task, which highlight the performance of both the approaches for a given input.

Scenario where MLPS works better than ML

1. ***Context*** : $< Start >$
Input : How much money do you spend daily on extras like a coffee or treat?
Ground Truth : personal-related-inquiry
MLPS : personal-related-inquiry
ML : task-related-inquiry

2. ***Context*** : .60 still sounds good to me. Lets leave it at that.
Input : I also want to assure you that Save the Children Makes huge impact on childrens lives internationally. They are extremely professional and your donation will go to a trustable fund.
Ground Truth : credibility-appeal
MLPS : credibility-appeal
ML : logical-appeal

3. ***Context*** : I wish there was a long-term solution to these problems.
Input : We all do, but for now, there are children in need and this organization does amazing work.
Ground Truth : logical-appeal
MLPS : logical-appeal
ML : credibility-appeal

4. ***Context*** : Hi! Doing good. How are you?
Input : I was wondering if I could talk to you about donating to Save the Children today?
Ground Truth : source-related-inquiry
MLPS : source-related-inquiry
ML : task-related-inquiry

5. ***Context*** : I could donate 10 cents. I wish I could more but I am trying to pay my bills with what I make here.

Input : The children will thank you, do you think you can do a little more?
Ground Truth : non-strategy
MLPS : non-strategy
ML : task-related-inquiry

Scenario where ML works better than MLPS

1. ***Context*** : We sponsor a child in El Salvador, we have been going it a number of years.
   Input : I donate money to save the children.
   Ground Truth : personal-story
   MLPS : self-modeling
   ML : personal-story

2. ***Context*** : I spend about 5 dollars a day, you?
   Input : Do you think that amount of money would make a difference in a needy child's life?
   Ground Truth : task-related-inquiry
   MLPS : personal-related-inquiry
   ML : task-related-inquiry

3. ***Context*** : I will donate 10 cents of my 30 cent payment. You should type the same thing and if perhaps if you know anything about the charity, share it?
   Input : This is a great charity and I will match you .10 cent payment.
   Ground Truth : self-modeling
   MLPS : non-strategy
   ML : self-modeling

4. ***Context*** : I'll donate but not that much
   Input : That's fine any amount helps.
   Ground Truth : foot-in-the-door
   MLPS : logical-appeal
   ML : foot-in-the-door

6 Conclusion

We summarize the conclusions of our work as: Pre-trained languages models like BERT may prove useful for natural language understanding of persuasion strategies even when data is scarce and imbalanced. Multilabel training which enforces a structure on the persuasion strategy class labels can help improve performance. A cost function based only on probabilistic structure was observed to provide the best performance. Probabilistic structure, even when used only during inference time, can provide competitive performance for coarser labels, which were not included in training. The performance gains due to MLPS were even more significant for classification of Persuadee's responses. MLPS offers more evenly distributed benefit for all the classes as compared to ML which can be more biased towards certain classes.

References

Siddhartha Banerjee, Cem Akkaya, Francisco Perez-Sorrosal, and Kostas Tsioutsiouliklis. 2019. Hierarchical transfer learning for multi-label text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6295–6300.

Timothy W Bickmore, Dina Utami, Robin Matsuyama, and Michael K Paasche-Orlow. 2016. Improving access to online health information with conversational agents: a randomized controlled experiment. *Journal of medical Internet research*, 18(1).

Anveshi Charuvaka and Huzefa Rangwala. 2015. Hiercost: Improving large scale hierarchical classification with cost sensitive learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 675–690. Springer.

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.

Pavlos Efraimidis and Paul Spirakis. 2008. *Weighted Random Sampling*, pages 1024–1027. Springer US, Boston, MA.

Siddharth Gopal and Yiming Yang. 2013. Recursive regularization for large-scale classification with hierarchical and graphical dependencies. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 257–265.

Arthur C Graesser, Haiying Li, and Carol Forsyth. 2014. Learning by communicating in natural language with conversational agents. *Current Directions in Psychological Science*, 23(5):374–380.

Christopher Thomas Hidey and Kathleen McKeown. 2018. Persuasive influence detection: The role of argument sequencing. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Simon Keizer, Markus Guhe, Heriberto Cuayáhuitl, Ioannis Efstathiou, Klaus-Peter Engelbrecht, Mihai Dobre, Alex Lascarides, Oliver Lemon, et al. 2017. Evaluating persuasion strategies and deep reinforcement learning methods for negotiation dialogue agents. ACL.

Kamran Kowsari, Donald E Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, Matthew S Gerber, and Laura E Barnes. 2017. Hdltex: Hierarchical deep learning for text classification. In *2017 16th IEEE international conference on machine learning and applications (ICMLA)*, pages 364–371. IEEE.

Yu Li, Kun Qian, Weiyan Shi, and Zhou Yu. 2019. End-to-end trainable non-collaborative dialog system. *arXiv preprint arXiv:1911.10742*.

Ewa Luger and Abigail Sellen. 2016. Like having a really bad pa: the gulf between user expectation and experience of conversational agents. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 5286–5297. ACM.

Piero Molino, Huaixiu Zheng, and Yi-Chia Wang. 2018. Cota: Improving the speed and accuracy of customer support through ranking and deep networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 586–595.

Gaku Morio, Ryo Egawa, and Katsuhide Fujita. 2019. Revealing and predicting online persuasion strategy with elementary units. 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 6275–6280.

Vitobha Munigala, Abhijit Mishra, Srikanth G Tamilselvam, Shreya Khare, Riddhiman Dasgupta, and Anush Sankaran. 2018. Persuaide! an adaptive persuasive text generation system for fashion domain. In *Companion Proceedings of the The Web Conference 2018*, pages 335–342.

Ankit Pal, Muru Selvakumar, and Malaikannan Sankarasubbu. 2020. Multi-label text classification using attention-based graph neural network. *arXiv preprint arXiv:2003.11644*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. 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 8024–8035. Curran Associates, Inc.

Mayur Patidar, Puneet Agarwal, Lovekesh Vig, and Gautam Shroff. 2018. Automatic conversational helpdesk solution using seq2seq and slot-filling models. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1967–1975.

Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 World Wide Web Conference*, pages 1063–1072.

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.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

Anthony Rios and Ramakanth Kavuluru. 2018. Few-shot and zero-shot multi-label learning for structured label spaces. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2018, page 3132. NIH Public Access.

Weiyan Shi, Xuewei Wang, Yoo Jung Oh, Jingwen Zhang, Saurav Sahay, and Zhou Yu. 2020. Effects of persuasive dialogues: Testing bot identities and inquiry strategies. *arXiv preprint arXiv:2001.04564*.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *Proceedings of the 25th international conference on world wide web*, pages 613–624.

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.

Xuewei Wang, Weiyan Shi, Richard Kim, Yoojung Oh, Sijia Yang, Jingwen Zhang, and Zhou Yu. 2019. Persuasion for good: Towards a personalized persuasive dialogue system for social good. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5635–5649, Florence, Italy. Association for Computational Linguistics.

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, Lukasz 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. *CoRR*, abs/1609.08144.

Changdong Xu and Xin Geng. 2019. Hierarchical classification based on label distribution learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5533–5540.

Diyi Yang, Jiaao Chen, Zichao Yang, Dan Jurafsky, and Eduard Hovy. 2019. Let's make your request more persuasive: Modeling persuasive strategies via semi-supervised neural nets on crowdfunding platforms. 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 3620–3630.

Seunghak Yu, Giovanni Da San Martino, and Preslav Nakov. 2019. Experiments in detecting persuasion techniques in the news. *arXiv preprint arXiv:1911.06815*.

Multi-Turn Target-Guided Topic Prediction with Monte Carlo Tree Search

Jingxuan Yang, Si Li[*] and Jun Guo
School of Artificial Intelligence, Beijing University of Posts and Telecommunications
`{yjx, lisi, guojun}@bupt.edu.cn`

Abstract

This paper concerns the problem of topic prediction in target-guided conversation, which requires the system to proactively and naturally guide the topic thread of the conversation, ending up with achieving a designated target subject. Existing studies usually resolve the task with a sequence of single-turn topic prediction. Greedy decision is made at each turn since it is impossible to explore the topics in future turns under the single-turn topic prediction mechanism. As a result, these methods often suffer from generating sub-optimal topic threads. In this paper, we formulate the target-guided conversation as a problem of *multi-turn topic prediction* and model it under the framework of Markov decision process (MDP). To alleviate the problem of generating sub-optimal topic thread, Monte Carlo tree search (MCTS) is employed to improve the topic prediction by conducting long-term planning. At online topic prediction, given a target and a start utterance, our proposed MM-TP (MCTS-enhanced MDP for Topic Prediction) firstly performs MCTS to enhance the policy for predicting the topic for each turn. Then, two retrieval models are respectively used to generate the responses of the agent and the user. Quantitative evaluation and qualitative study showed that MM-TP significantly improved the state-of-the-art baselines.

1 Introduction

Although impressive efforts have been made to integrate background knowledge into the conversation systems (Fang et al., 2018; Qin et al., 2019; Liu et al., 2018), existing open-domain conversation systems still suffer from creating generic response (Yi et al., 2019) and struggle to perform engaging conversations (Ram et al., 2018). Moreover, there exists a strong demand in real-world applications to integrate the goals and strategies into the open-domain conversation systems, to make them achieve some specific goals such as recommending an item or accomplishing nursing goals. Faced with these problems, target-guided open-domain conversation (Tang et al., 2019; Sevegnani et al., 2021) has attracted increasing research attentions.

[*] Corresponding author

Topic	Response *Target : job*
Turn: 1	A: Hi, how are you today? U: Pretty good. Chilling and listening to some music.
Turn: 2 *listen*	A: Cool, I'm *listening* to some old rock music. U: Awesome, which kind of music is your favorite?
Turn: 3 *music*	A: Hank Williams. I grew up on country *music*. U: Cool. Not much of a singer.
Turn: 4 *job*	A: When I was a child, I had to get a *job* to help my parents.

Figure 1: An example conversation conducted between the single-turn topic prediction conversational system agent (A) and user (U).

Different from traditional open-domain conversation, target-guided open-domain conversation requires the system to proactively and naturally guide the conversational thread, and end up with recommending a target item or mentioning a target word. Existing studies (Tang et al., 2019; Qin et al., 2020; Zhong et al., 2020) usually resolve the task with a sequential of single-turn topic predictions and response generations. At each turn, the model firstly selects a topic from the candidate topic set based on the history context, and then retrieves response according to the selected topic. Since the single-turn topic prediction mechanism has no ability to plan the topics in the future turns, greedy decision has to be made at each turn. As a result, these methods usually suffer from generating sub-optimal topic threads.

Figure 1 illustrates an example conversation between user and the Kernel agent (Tang et al., 2019), which utilizes single-turn topic prediction model to select topics. At the third turn, the sub-optimal topic "music" was selected. Though it is strongly relevant to the topic “listen” in the second turn, it is irrelevant to the final target topic “job”. The example verified that the greedy decisions in the single-turn topic prediction cannot naturally guided the conversation to achieve the target.

To deal with the issue, we propose to formulate target-guided conversation as a multi-turn topic prediction problem, and model it with Markov decision process (MDP). In the MDP, the environment is responsible for collecting the conversational history as the states, and the conversational system agent is responsible for selecting action as topic for each turn. Inspired by the reinforcement learning method of AlphaGo Zero (Silver et al., 2017), we utilize Monte Carlo tree search (MCTS) to make a long-term planning by considering the topics in the future turns and then generate topic for the current turn. Given a pre-defined target topic and a randomly selected start utterance, the proposed model, referred to as MM-TP (MCTS-enhanced MDP for Topic Prediction), iteratively generates topic sequence and guides the conversation to achieve the target topic. At each turn, MCTS is firstly utilized to enhance the raw policy and predict the topic of this turn. Two retrieval models are then respectively employed to generate the responses of the agent and the user. In this way, the problem of generating sub-optimal topic threads could be alleviated by the MCTS at a certain extent.

We conducted experiments on two popular target-guided open-domain conversation benchmarks. Quantitative results show that MM-TP outperformed the state-of-the-art baselines by achieving the target more accurately and providing more smooth topic transition. Qualitative study also show that our MM-TP improved baseline methods by making long-term planning of the topics. The major contributions of the paper are three-fold:

- To the best of our knowledge, it is the first time that the target-guided conversation is formalized as a multi-turn topic prediction problem and solved under the framework of MDP.
- We adapt the traditional MCTS for the target-guided open-domain conversation, to alleviate the sub-optimal topic threads generation problem by performing long-term planning.
- The proposed MM-TP model outperformed the baseline methods in terms of achieving the targets more accurately and making more smoothly topic transition.

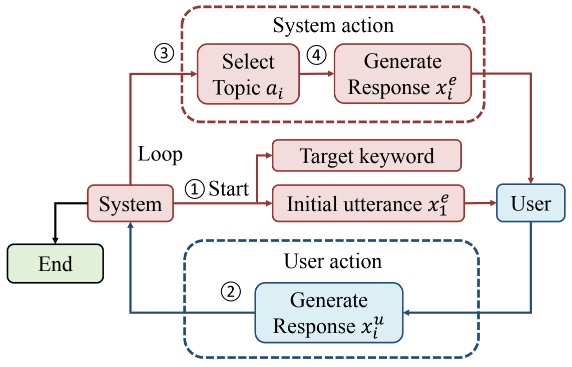


Figure 2: Workflow of Multi-turn Target-guided Open-domain Conversation

2 Task Definition: Multi-turn Target-guided Topic Prediction

As shown in Figure 2, a multi-turn target-guided open-domain conversation system starts with randomly selecting a specific target topic and the start utterance (step 1) by the simulator. The user generates an appropriate response (step 2). Then, the system repeats several conversational turns before achieving the ends. At each turn, the system first accesses to conversational history utterances and predicts a topic (step 3) satisfying both transition smoothness and target achievement. Then the agent and user generate responses respectively according to the predicted topic (step 4 and step 2). During the conversation, the target word is only presented to the agent and is unknown to user. The system consists of two components which are topic prediction module and response generation module.

Formally, let’s use $\mathcal{A}$ and $\mathcal{X}$ to denote the sets of candidate target topics and responses, respectively. Following the practices in (Tang et al., 2019; Qin et al., 2020), each target topic $a \in \mathcal{A}$ is defined as a word/phrase (i.e., an entity name or a common noun), and the candidate utterance set $\mathcal{X}$ is derived from the PersonaChat corpus (Zhang et al., 2018). Suppose that the agent e starts a conversation (1st turn) with utterance $\boldsymbol{x}_1^e$ and its target topic is a^*. The user retrieval model $\mathcal{G}^u$ generates a response $\boldsymbol{x}_1^u$. Then, at each turn $i \in \{2, \cdots, m\}$, the topic prediction module takes previous utterance context $\mathrm{X}_i = \{\boldsymbol{x}_1^e, \boldsymbol{x}_1^u, \cdots, \boldsymbol{x}_{i-1}^u\}$ as input and outputs the predicted topic a_i. Then, the retrieval model $\mathcal{G}^u$ for user u and $\mathcal{G}^e$ for system agent e select a response from the candidate set $\mathcal{X}$ respectively. As an appropriate measurement of the success rate, the target is regarded as achieved when the predicted topic a_m is similar enough to the target topic a^*.

3 The Proposed Model: MM-TP

3.1 Model overview

In this work, we focus on formulating Multi-turn Target-guided Topic Prediction as an MDP and utilizing the MCTS-enhanced policy to select the topic for each turn with a long-term planning. For the response generation process, we utilize the simulator constructed in (Tang et al., 2019), and employ kernel-based retrieval model as $\mathcal{G}^e$ and conventional retrieval model as $\mathcal{G}^u$ to generate responses by agent and user respectively. Figure 3 illustrates the architecture of the proposed MCTS-enhanced MDP for Topic Prediction (MM-TP) model. Given the target word a^* and the start utterance $\boldsymbol{x}_1^e$, our model iterates several turns for guiding the conversation thread. For each turn, MM-TP first applies MCTS to select topic for the current turn, and then utilizes the retrieval models $\mathcal{G}^e$ and $\mathcal{G}^u$ to generate agent and user response respectively.

3.2 MDP formulation of Multi-turn Target-guided Topic Prediction

MM-TP models the Multi-turn Target-guided Topic Prediction as a process of sequential decision making with MDP, in which each time step corresponds to a conversational turn. The states, actions, transition function, rewards, value function and policy function of the MDP are defined as:

States $\mathcal{S}$: The state of each turn is defined as a tuple $s_t = [\mathrm{X}_t = \{\boldsymbol{x}_1^e, \boldsymbol{x}_1^u, \cdots, \boldsymbol{x}_{t-1}^u\}, \mathrm{Y}_t = \{a_1, \ldots, a_{t-1}\}]$ where X_t is the sequence of contextual utterances and Y_t is the sequence of predicted topics in previous $t-1$ turns. For the second turn, the state is initialized as $s_2 = [\{\boldsymbol{x}_1^e, \boldsymbol{x}_1^u\}, \emptyset]$, where $\{\boldsymbol{x}_1^e, \boldsymbol{x}_1^u\}$ denotes the randomly selected start utterance and the first response of user. $\emptyset$ denotes the empty topic sequence.

Actions $\mathcal{A}$: At each turn t, the $\mathcal{A}(s_t) \subseteq \mathcal{Y}$ is the set of actions the agent can choose from, which means the action $a_t \in \mathcal{A}(s_t)$ is the predicted topic $a_t \in \mathcal{Y}$ for the current turn.

Transition function T: The transition function $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is defined as: $s_{t+1} = T(s_t, a_t) = T([\mathrm{X}_t, \mathrm{Y}_t], a_t) = [\mathrm{X}_{t+1}, \mathrm{Y}_t \oplus a_t]$, where $\oplus$ appends the selected action a_t to Y_t. At each turn t, based on state s_t, the system predicts a topic a_t for this turn, moves to the turn $t+1$ and transits the state to the next state s_{t+1}: first, the conversational utterance context X_t is updated by appending the generated agent and user responses; second, the system adds the predicted topic to the end of Y_t, outputting a new topic sequence.

Rewards $\mathcal{R}$: The reward is defined to reflect: (1) target achievement $\mathcal{R}_{ta}$: we calculate the similarity between the predicted topic of each turn and the target to determine whether the topic has achieved the target; (2) local smoothness $\mathcal{R}_{ls}$: we calculate the average WordNet similarity between topics of adjacent turns to measure the topic transition smooth; (3) target similarity $\mathcal{R}_{ts}$: we calculate the similarity difference between the adjacent topics and the target, to make the predicted topic in each turn is more similar to that in the preceding turns.The overall reward is defined as the weighted summation these three parts as:

$$\mathcal{R} = \alpha \cdot \mathcal{R}_{ta} + \beta \cdot \mathcal{R}_{ls} + \gamma \cdot \mathcal{R}_{ts}$$

where α, β, γ are weight parameters for three kinds of rewards respectively.

Value function V: The value function V is a scalar evaluation which is learned to estimate the the quality of topic assignments and fit the real evaluation measure. In this work, we utilize a hierarchical GRU network to map the context X_t to a real vector, and then define the value function as a nonlinear transformation of the weighted sum of the MLP's outputs $g(s)$ and the current candidate action in one-hot representation a_t as:

$$\mathrm{V}(s) = \sigma(\langle \mathbf{W}_v g(s), a_t \rangle),$$

where $\mathbf{W}_v \in \mathbb{R}^{|\mathcal{A}(s)| \times |g(s)|}$ is the weight vector to be learned during training. $\langle \cdot, \cdot \rangle$ is dot product operation, and $\sigma(\cdot)$ is the nonlinear sigmoid function. The context state $g(s)$ is obtained as:

$$g(s) = \mathrm{MLP}(l(s)),$$

$$l(s) = [\mathrm{HierarchalGRU}(\mathrm{X}_t)].$$

The hierarchical GRU network takes in a sequence of contextual utterances $\mathrm{X}_t = \{\boldsymbol{x}_1^a, \boldsymbol{x}_1^u, \cdots, \boldsymbol{x}_{t-1}^u\}$ and utilizes the word-level GRU to encode each utterance and output a representation of the utterance. Then, the sequence of utterance representations are fed into a utterance-level GRU for obtaining a conversational context representation $l(s)$.

Policy function $\mathbf{p}$: The policy function $\mathbf{p}(s)$ takes the context representation $g(s)$ as input and outputs a distribution over all possible actions $a \in \mathcal{A}(s)$, in which each element represents the probability of selecting this keyword as:

$$p(a|s) = \mathrm{softmax}(\mathrm{U}_p g(s)),$$

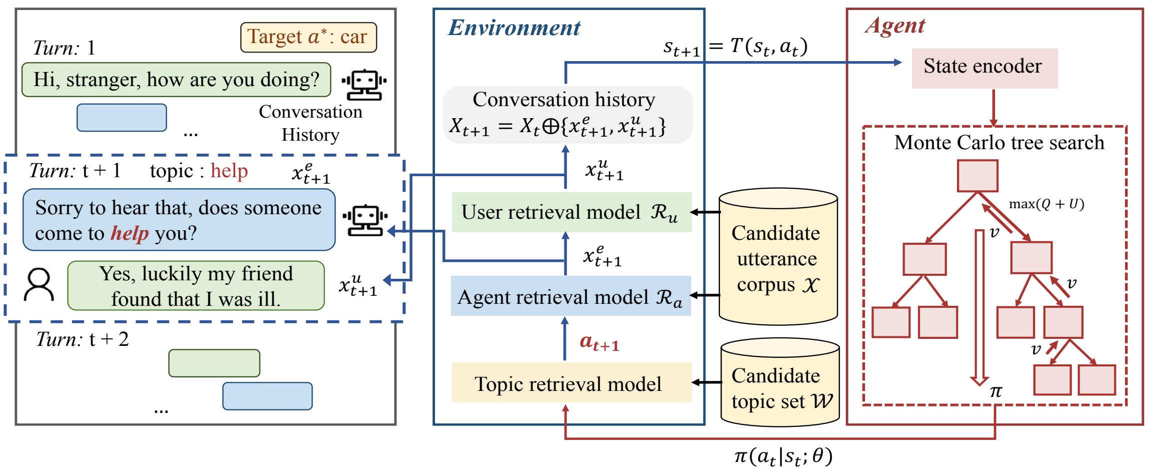

Turn: 1
Target a^*: car
Hi, stranger, how are you doing?
Conversation History
Turn: t + 1
topic : help
x^e_{t+1}
Sorry to hear that, does someone come to help you?
Yes, luckily my friend found that I was ill.
x^u_{t+1}
Turn: t + 2
Environment
$s_{t+1} = T(s_t, a_t)$
Conversation history
$X_{t+1} = X_t \oplus \{x^e_{t+1}, x^u_{t+1}\}$
User retrieval model $\mathcal{R}_u$
Agent retrieval model $\mathcal{R}_a$
a_{t+1}
Topic retrieval model
Candidate utterance corpus $\mathcal{X}$
Candidate topic set $\mathcal{W}$
Agent
State encoder
Monte Carlo tree search
max(Q + U)
v
π
$\pi(a_t|s_t;\theta)$

Figure 3: Overview of MM-TP. The agent guides the conversation to achieve the target by multi-turn topic prediction, which is formulated as an MDP process. For each turn, the model first encodes the previous conversation context, and then updates the search policy π to predict the topic for the next turn.

where $\mathrm{U}_p \in \mathbb{R}^{|\mathcal{A}(s)|\times|g(s)|}$ is the parameter. The policy function is obtained as:

$$\mathbf{p}(s) = \left\langle p(a_1|s), \cdots, p(a_{|\mathcal{A}(s)|}|s)\right\rangle. \quad (1)$$

At online topic sequence prediction stage, the environment collects the conversational history utterances and the predicted topic sequence as the states s_t, and then pass them to the system agent. Once received states, the agent firstly encodes them through the hierarchical GRUs and then performs MCTS to update the search policy $\boldsymbol{\pi}$, guided by the policy function $\mathbf{p}$ and value function V. The updated policy $\boldsymbol{\pi}$ is used to select action as the predicted topic for this turn.

3.3 Improve raw policy with MCTS

Predicting the topic for each turn with the raw policy $\mathbf{p}$ (Eq. 1) only considers the past states and often leads to sub-optimal results. To alleviate the issue, we conduct lookahead search with MCTS for each turn and output a improved search policy $\boldsymbol{\pi}$ to select the topic.

Specifically, MCTS takes a root node s_{R}, value function V and policy function $\mathbf{p}$ as input, and iterates K times to output a improved search policy $\boldsymbol{\pi}$ which selects a topic for the current turn. Each tree node corresponds to an MDP state. Each edge $e(s, a)$ stores an action value $\mathrm{Q}(s, a)$, visit count $\mathrm{N}(s, a)$ and prior probability $\mathrm{P}(s, a)$. For each iteration, the raw policy $\mathbf{p}$ is improved by four steps: (1) Selection: Each iteration starts from the root node s_{R} and iteratively selects a topic for each turn to maximize action value plus a bonus as $a_t = \arg\max_a(\mathrm{Q}(s_t, a) + \lambda\mathrm{U}(s_t, a))$, where $\lambda \geq 0$ is the tradeoff coefficient, and the bonus $\mathrm{U}(s_t, a) = p(a|s_t)\frac{\sqrt{\sum_{a'\in\mathcal{A}(s_t)}\mathrm{N}(s_t,a')}}{1+\mathrm{N}(s_t,a)}$ is proportional to the prior probability but decays with repeated visits to encourage exploration. (2) Evaluation and expansion: When the traversal reaches a leaf node s_{L}, the node is evaluated with the value function V. Then, the leaf node s_{L} is expanded by constructing edge from it to the node $\mathrm{T}(s_{\mathrm{L}}, a)$, corresponding to each action $a \in \mathcal{A}(s_t)$. (3) Back-propagation and update: At the end of evaluation, the action values and visit counts of all traversed edges are updated, while the prior probability $\mathrm{P}(s, a)$ is kept unchanged. (4) Calculate the improved search policy: After iterating K times, the improved search policy $\pi(a|s_{\mathrm{R}})$ corresponds to each $a \in \mathcal{A}(s_{\mathrm{R}})$ for the current root node s_{R} is calculated based on the visit counts $\mathrm{N}(s_{\mathrm{R}}, a)$ of the edges starting from s_{R}. The details of MCTS process is described in Algorithm 1.

3.4 Model training and inference

MM-TP has some parameters $\boldsymbol{\Theta}$ to learn including $\mathbf{W}_v, \mathbf{W}_g, \mathrm{U}_p, b_g$ and parameters in hierarchical GRUs. Suppose we are given N target topics and ground-truth topic threads that achieved the corresponding target topics: $\mathcal{D} = \{(a^{*(n)}, \mathrm{Y}^{(n)})\}_{n=1}^{N}$. Firstly, the parameters $\boldsymbol{\Theta}$ of the model are initialized to random weights in $[-1, 1]$. Then for each sample $(a^*, \mathrm{Y}) \in \mathcal{D}$, a topic sequence is predicted as: for each turn, the MCTS is executed and a topic a_t is selected by the search policy $\boldsymbol{\pi}_t$. The topic prediction process terminates after m turns,

Algorithm 1 TreeSearch

```
Input: root $s_{\mathrm{R}}$, Value function V, policy function p, search times K
1:  for $k = 0$ to K $- 1$ do
2:      $s_{\mathrm{L}} \leftarrow s_{\mathrm{R}}$
3:      {Selection}
4:      while $s_{\mathrm{L}}$ is not a leaf node do
5:          $a \leftarrow \arg\max_a(\mathrm{Q}(s_t, a) + \lambda\mathrm{U}(s_t, a))$
6:          $s_{\mathrm{L}} \leftarrow$ child node pointed by $(s_{\mathrm{L}}, a)$
7:      end while
8:      {Evaluation and expansion}
9:      $v \leftarrow \mathrm{V}(s_{\mathrm{L}})$ {simulate $v$ with V}
10:     for all $a \in \mathcal{A}(s_{\mathrm{L}})$ do
11:         Expand $e$ to $s = [s_{\mathrm{L}}.\mathrm{X}_{t+1}, \mathrm{Y}_t \oplus \{a\}]$
12:         $e.\mathrm{P} \leftarrow p(a|s_{\mathrm{L}}); e.\mathrm{Q} \leftarrow 0; e.\mathrm{N} \leftarrow 0$
13:     end for
14:     {Back-propagation}
15:     while $s_{\mathrm{L}} \neq s_{\mathrm{R}}$ do
16:         $s \leftarrow$ parent of $s_{\mathrm{L}}$
17:         $e \leftarrow$ edge from $s$ to $s_{\mathrm{L}}$
18:         $e.\mathrm{Q} \leftarrow \frac{e.\mathrm{Q} \times e.\mathrm{N} + v}{e.\mathrm{N}+1}$
19:         $e.\mathrm{N} \leftarrow e.\mathrm{N} + 1$; $s_{\mathrm{L}} \leftarrow s$
20:     end while
21: end for
22: for all $a \in \mathcal{A}(s_{\mathrm{R}})$ do
23:     $\pi(a|s_{\mathrm{R}}) \leftarrow \frac{e(s_{\mathrm{R}},a).\mathrm{N}}{\sum_{a' \in \mathcal{A}(s_{\mathrm{R}})} e(s_{\mathrm{R}},a').\mathrm{N}}$
24: end for
25: return $\boldsymbol{\pi}$
```

and a topic sequence $\hat{\mathrm{Y}} = \{a_1, \ldots, a_m\}$ is outputted. The overall evaluation metric r of $\hat{\mathrm{Y}}$ is calculated according to the success rate of the target achievement. The data generated at each turn $\mathrm{E} = \{(s_t, \pi_t)\}_{t=1}^{m}$ and the reward $\mathcal{R}$ are utilized as the signal for adjusting the value function. The training objective is to minimize the error between the predicted value $\mathrm{V}_{(s_t)}$ and evaluation metric r, and to maximize the similarity between the raw policy $\mathbf{p}_{(s_t)}$ and the search policy $\boldsymbol{\pi}_t$ as:

$$l(\mathrm{E}, r) = \sum_{t=1}^{|\mathrm{E}|}((\mathrm{V}_{(s_t)} - r)^2 + \sum_{a \in \mathcal{A}(s_t)} \pi_t(a|s_t) \log \frac{1}{p(a|s_t)}). \tag{2}$$

Algorithm 2 shows the details of the training process. The inference process of the MM-TP model is similar to the training stage. Given the selected target topic, the state is initialized as $s_2 = [\mathrm{X}_1, \mathrm{Y}_1]$. For each turn $t \in \{2, \cdots, \mathrm{m}\}$, the agent receives

Algorithm 2 Train MM-TP model

```
Input: Labeled data D, learning rate $\eta$, search time K, pre-defined number of turn $m$
1:  Initialize $\Theta$ as random values in $[-1, 1]$
2:  repeat
3:      for all $(\mathrm{X}, \mathrm{Y}) \in \mathrm{D}$ do
4:          $s_2 = [\mathrm{X}_1, \mathrm{Y}_1]; \mathrm{E} \leftarrow \emptyset$
5:          for $t = 1$ to $m$ do
6:              $\pi \leftarrow$ TreeSearch $(s, \mathrm{V}, \boldsymbol{\pi}, \mathrm{K})$
7:              $a = \arg\max_{a \in \mathcal{A}_{(s)}} \pi(a|s)$
8:              $\mathrm{E} \leftarrow \mathrm{E} \oplus \{(s, \pi)\}$
9:              $s \leftarrow [s.\mathrm{X}_{t+1}, s.\mathrm{Y}_t \oplus \{a\}]$
10:         end for
11:         $r \leftarrow$ Metric$(\mathrm{Y}, s.\mathrm{Y}_m)$
12:         $\Theta \leftarrow \Theta - \eta\frac{\partial \ell(\mathrm{E},r)}{\partial \Theta}$ {see $\ell$ in Eq. 2}
13:     end for
14: until converge
15: return $\Theta$
```

the state $s_t = [\mathrm{X}_t, \mathrm{Y}_t]$ and updates the search policy $\boldsymbol{\pi}$ with MCTS. Then, MM-TP selects an action a_t for this turn and moves to the next turn whose state becomes $s_{t+1} = [\mathrm{X}_{t+1}, \mathrm{Y}_{t+1}]$.

3.5 Implementation details

We adapt the MCTS algorithm according to our task. Following existing practice (Tang et al., 2019; Qin et al., 2020), in order to guide the topic thread to achieve the target keyword, we shrink the action space in each conversational turn. Specifically, we mask the candidate topics which have been selected in preceding turns, and the candidates that are not as similar to the target as the topics in preceding turns. The tree nodes corresponding to these masked nodes thus will not be achieved during the update process of search policy $\boldsymbol{\pi}$. Moreover, we also load the parameters of pre-trained single-turn topic prediction model (Tang et al., 2019) to initialize the policy and value network, and the parameters are also updated during training process.

4 Experiments

4.1 Experimental settings

Datasets: We evaluated the performance of MM-TP on two popular conversation benchmarks: Target-Guided PersonaChat dataset (TGPC) and Chinese Weibo Conversation dataset (CWC). The TGPC dataset (Tang et al., 2019) is derived from the PersonaChat corpus which covers a abroad range of topics. Following (Tang et al., 2019),

Dataset	CWC		TGPC	
	Train	Test	Train	Test
#Conversations	824,742	45,763	8,939	500
#Utterances	1,104,438	60,893	101,935	5,317
#Keyword types	1,760	1,760	2,678	1,571

Table 1: Statistics of training and test sets on two conversation benchmarks.

we take 500 conversations with relatively frequent keywords as the test set. The CWC dataset (Qin et al., 2020) is a Chinese conversational dataset that derived from corpus crawled from Sina Weibo paltform. It matches the real-word scenarios better and more efficient for the model to learn dynamic topic transition. The statistics of these two benchmarks are reported in Table 1.

Baselines: Existing target-guided open-domain conversation systems are used as baselines: (1) Retrieval (Wu et al., 2017) is a conventional retrieval-based chitchat system that used to provide reference performance in terms of different metrics; (2) Retrieval-Stgy (Tang et al., 2019) which augments the above Retrieval system with the target-guided strategy and permits the system to retrieve a response containing more than one keyword; (3) PMI (Tang et al., 2019) which constructs a keyword pairwise matrix, and calculates the association between keywords by pointwise mutual information; (4) Neural (Tang et al., 2019) which utilizes a neural network to encode the conversation history and then employs a prediction layer to select a keyword for the next turn. (5) Kernel (Tang et al., 2019) which firstly measures the similarity between the current keyword and candidate keywords, and then utilizes a kernel layer to predict the candidate probability distribution; (6) DKRN (Qin et al., 2020) which uses the semantic knowledge relations among candidate keywords to mask the candidates uncorrelated to the conversational history.

Training Details: Following (Tang et al., 2019; Qin et al., 2020), we used GloVe (Pennington et al., 2014) to initialize word embeddings for English conversation corpus TGPC and Baidu Encyclopedia Word2Vec (Li et al., 2018) to initialize word embeddings for Chinese conversation corpus CWC. The number of conversational turns m was set as 8. The hierarchical GRU network utilized a hidden layer of 200 units. We used the AdaGrad (Duchi et al., 2011) optimizer to update the parameters during the training process, with a learning rate η

Model	TGPC		CWC	
	Succ.(%)	Turns	Succ.(%)	Turns
Retrieval	7.16	4.17	0	-
Retrieval-Stgy	47.80	6.7	44.6	7.42
PMI	35.36	6.38	47.4	5.29
Neural	54.76	4.73	47.6	5.16
Kernel	62.56	4.65	53.2	4.08
DKRN	89.0	5.02	84.4	4.20
MM-TP	**91.23**	**4.82**	**86.3**	**4.15**

Table 2: Results of our MM-TP and baseline conversation systems in terms of successful rate ("Succ.%") and average turns of target achievement ("Turns").

as 0.001. The search time K in MCTS was set to 1600, and the tradeoff coefficient λ was set to 80.0. Two retrieval systems $\mathcal{G}_e$ and $\mathcal{G}_u$ were implemented with the toolkit Texar (Hu et al., 2019).

4.2 Self-play simulation evaluation

We first conducted simulation-based evaluation of our MM-TP and baseline systems in the multi-turn target-guided conversation setup. Same as (Tang et al., 2019; Qin et al., 2020), we employed the conventional retrieval system to play the role of human. The baseline models and our MM-TP played the role of system agent aiming to guide the conversation to achieve the target topic. During the training process, we generated the ground-truth topic threads by iteratively appending the keyword sequences from the consecutive single-turn keywords prediction samples in existing work (Tang et al., 2019). In the testing phrase, the simulator randomly selected a target from the candidate topic set and the start utterance from the corpus. The experiment was evaluated by measuring the success rate of achieving the target (**Succ.%**), and the average number of turns used to reach the target (**Turns**). The target topic is considered as achieved when any item of the predicted topic sequence takes a similarity score with the target higher then 0.9, measured by WordNet (Fellbaum and Miller, 1998).

Table 2 reports the results of our MM-TP as well as the baselines on TGPC and CWC. From the results, we can see that our proposed model outperformed the baselines in terms of success rate on both of the datasets. We attribute this to that MM-TP takes a long-term planning to select the topic by considering the topics in next several turns. Moreover, the average turns of MM-TP to achieve the target is comparable to baseline methods since

Model	**Smoothness**
Retrieval-Stgy	0.08
PMI	0.21
Neural	0.25
Kernel	0.23
DKRN	0.31
MM-TP	**0.35**

Table 3: Results of MM-TP and baseline methods in terms of transition smoothness.

the long-term planning explores an optimized topic thread to achieve the target.

4.3 Effects of Monte Carlo tree search

The search policy $\boldsymbol{\pi}$ usually performs better than the raw policy $\mathbf{p}$ since MCTS is employed to consider the topics in next several turns. Except the policies, the value function V can also be used to select topic at each turn. To explore the effectiveness of these three components, we applied them to predict the topic sequence on the test set respectively after every 20 training epochs during the online training phrase, and records the average success rate of target achievement. Figure 4 illustrates the success rate curves of the raw policy $\mathbf{p}$, search policy $\boldsymbol{\pi}$, and value function V. We can see that: (1) The topic sequences generated by the search policy $\boldsymbol{\pi}$ achieves higher success rate of target achievement than that generated by the raw policy $\mathbf{p}$, which demonstrates that MCTS improved the raw policy. (2) The results predicted by both $\boldsymbol{\pi}$ and $\mathbf{p}$ are better than results predicted by the value function V. The reason is that the raw policy $\mathbf{p}$ and value network V greedily select topic at each conversational turn, which makes the results are not as good as that predicted by the search policy $\boldsymbol{\pi}$. Moreover, the quality of topic assignment is not easy to estimate by value function.

4.4 Transition smoothness evaluation

We further explore how our MM-TP accomplishes transition smoothness, which is also an important objective of target-guided conversation for measuring how naturally the conversation is guided. We evaluate our proposed model and baseline methods in terms of transition smoothness. Specifically, the transition smoothness (**Smoothness**) of each model is calculated by the average WordNet information content similarity between topics in adjacent turns. Table 3 shows the results of transition

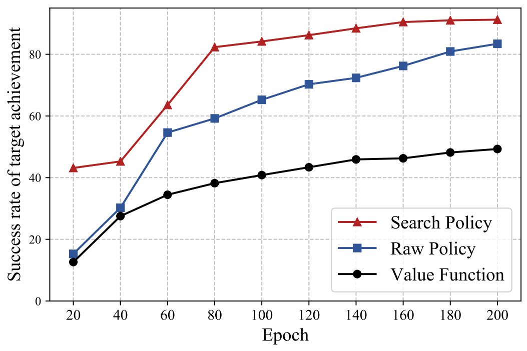


Figure 4: Success rate w.r.t. training process, when the actions were selected with raw policy $\mathbf{p}$, search policy $\boldsymbol{\pi}$, and value function V.

smoothness of our proposed MM-TP and baseline methods. We can see that MM-TP achieves higher transition smoothness compared with baseline systems. We contribute this to that the baseline methods are only constrained to select a topic at each turn which is strictly more closer to the target topic than those in preceding turn, while the transition smoothness between the topics in adjacent turns is overlooked. The proposed MM-TP improves these methods by modeling the transition smoothness between topics in adjacent turns as local rewards, and the performance of transition smoothness can be controlled by adjusting the weight parameter β of local smoothness.

4.5 Qualitative study

To dive a bit deeper and look at the performance of our MM-TP on topic sequence prediction, we compare the examples outputted by different conversational systems, and the results are shown in Figure 5. The three agents are given the same target topic and start utterance, and the task is considered as successful when the predicted topic is similar enough to the target. We can see that the Kernel agent (Tang et al., 2019) employs single-turn keyword prediction and utilizes the rule strategy to make the topic predicted at each turn is strictly closer to the target than topics selected in preceding turns. As a result, the topics predicted in adjacent turns are distantly related and leading to poor smooth transition. The DKRN agent (Qin et al., 2020) improves the Kernel agent by considering the relations between candidate keywords, and leads to better smooth transition. However, the method still suffers from predicting sub-optimal topics as it overlooks topics in next several turns. For example, in the conversation produced by DKRN, the top-

topic/score	Agent: Kernel Target: car
	A: Hi, stranger, how are you doing? U: Pretty good. Listening to some music.
sound/0.60	A: That sounds relaxing. I am about to settle down and read some poetry. U: What book? I hate reading.
truck/0.80	A: Well, former truck driver, like my dad, but now I just fix them. U: I've never been an artist type.
car/1.0	A: Me too, I also love to race cars.
	Agent: DKRN Target: car
	A: Hi, stranger, how are you doing? U: A bit stressed but alright.
play/0.56	A: I understand that. I got back from playing piano at my father's funeral. U: I am watching my favorite show.
game/0.66	A: I enjoy bingo, we have a game once a week. U: My hobby is watching movie.
music/0.69	A: Wonderful to spend time listening music and away from writing poems. U: Awesome, I also enjoy writing.
car/1.0	A: That is cool. The car is leased anyway so it isn't that impressive.
	Agent: MM-TG Target: car
	A: Hi, stranger, how are you doing? U: I feel a little uncomfortable, want to sleep.
sick/0.53	A: Sorry to hear, are you sick? U: Yes, I am just recovering from an asthma attack and cannot go out.
help/0.66	A: Sorry to hear that, does someone come to help you? U: Yes, luckily my friend found that I was ill.
car/1.0	A: Did he take you to the hospital by car?

Figure 5: Example conversations between user (U) and three different agents (A) with the same target topic (car) and start utterance.

ics are smoothly transitioned in first four turns but changed sharply from "music" to "car" in the fifth turn in order to achieve the target. In contrast, our MM-TP takes a smooth transition from the start topic to the target as "the stranger is *sick* and his friend *helps* him to hospital by *car*". We contribute this to that MM-TP searches the topic thread that will be selected in next several turns by MCTS and then makes prediction.

4.6 Error Analysis

Besides conducting self-play simulation evaluation and performing qualitative evaluation by case study, we also investigate some typical mistakes made by our MM-TP. For example, in Figure 6, the agent communicates with the user for eight turns and finally fail to achieve the target. We attribute this to that the model lacks some common sense knowledge (e.g., knowledge graph of topic) and attempt to select a near-synonym for the next turn. Specifically, according to the definition of Reward in MM-TP, the transition between topics of consecutive turns should satisfies *smoothness transition* and *target similarity*. However, as not any common sense knowledge are injected into our model, the search policy of MCTS is just trained to select a topic similar to that in the previous turn and more closer to the target. Whether the selected topic is logically related to the topic in the previous turn and can leading the topic thread to the target is overlooked.

topic/score	Agent: MM-TG Target: Baseball
	A: Hi, how are you today? U: Fine. Just finishing up my homework.
reading/0.13	A: Oh, Do you like reading ? U: Yes, I enjoy reading romantic stories.
writing/0.15	A: That sounds helpful for writing U: Yeah, but I do not good at writing articles
singing/0.16	A: Oh cool! I really like music and singing. U: That's great. My favorite singer is Taylor Swift.
dancing/0.24	A: I also love dancing, although I cannot dance well. U: Music is life. Dancing to anything with a beat is the best.
riding/0.26	A: Fine. Sometimes I also like riding in the countryside. U: That sounds relaxing. I love the outdoors.
swimming/0.29	A: Do you like swimming? U: I often walk along a long black river, but I cannot swim.
playing/0.55	A: I also like playing football on weekends. U: I hope I could paly with you.

Figure 6: Example errors made by MM-TG.

5 Related Work

Existing research of dialogue system can be broadly concluded as two categories, which are task-oriented dialogue systems and open-domain dialogue systems. Task-oriented dialogue system aims to accomplish some pre-defined goals (Lipton et al., 2018), conduct negotiation (Cao et al., 2018) or perform symmetric collaborations (He et al., 2017). Open-domain dialogue systems are designed to chat naturally with human and aiming to provide reasonable responses. Previous work make efforts to improve response generation by developing novel neural networks and training on large-scale corpus (Serban et al., 2017; Zhou et al., 2016, 2018). Although the promising progresses have been achieved, these chat-oriented dialogue systems still struggle to a set of limitations such as dull or inconsistent responses (Ram et al., 2018).

Due to these limitations, a novel task named target-guided open-domain conversaion was proposed, which requires the system to proactively and naturally guide the topic thread by integrating goals and strategies. Tang et al. (2019) for the first time introduced this task and employed a simple target-guided strategy to attain smooth topic transition by turn-level supervised learning. Qin et al. (2020) further improved this work by capturing semantic or factual knowledge relations among candidate topics through a dynamic knowledge routing network. However, both these methods employ single-turn supervised learning to predict the topic of each turn according to the human annotated topic sequence. Moreover, they only consider existing context and overlook the long-term planning of topics in next several turns.

Monte Carlo Tree Search (MCTS) enhanced MDP was firstly proposed in games (Silver et al., 2016; Schrittwieser et al., 2019; Silver et al., 2017) and has been applied in other fields such as diverse ranking (Feng et al., 2018), name entity recognition (Lao et al., 2019) and task-oriented conversation (Wang et al., 2020). In this paper, we apply MCTS in open-domain conversation to generate topic sequence which is utilized to guided the conversation thread to achieve the target.

6 Conclusion

In this paper, we formulate the target-guided conversation as a multi-turn topic prediction problem, and propose a novel approach called MM-TP to resolve this task. MM-TP formalizes the multi-turn topic prediction as sequential decision prediction problem, and models it with MDP. MCTS is used to improve the raw policy by making a long-term planning of topics in next several turns and then selecting a topic for the current turn. The model parameters are learned by reinforcement learning. Experimental results demonstrate that MM-TP outperformed existing baseline systems in terms of both the successful rate of achieving target and the topic transition smoothness.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61702047) and BUPT Excellent Ph.D. Students Foundation (No.CX2020305).

References

Kris Cao, Angeliki Lazaridou, Marc Lanctot, Joel Z. Leibo, Karl Tuyls, and Stephen Clark. 2018. Emergent communication through negotiation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159.

Hao Fang, Hao Cheng, Maarten Sap, Elizabeth Clark, Ari Holtzman, Yejin Choi, Noah A. Smith, and Mari Ostendorf. 2018. Sounding board: A user-centric and content-driven social chatbot. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 2-4, 2018, Demonstrations*, pages 96–100. Association for Computational Linguistics.

C Fellbaum and G Miller. 1998. *WordNet : an electronic lexical database*. MIT Press.

Yue Feng, Jun Xu, Yanyan Lan, Jiafeng Guo, Wei Zeng, and Xueqi Cheng. 2018. From greedy selection to exploratory decision-making: Diverse ranking with policy-value networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 125–134. ACM.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1766–1776. Association for Computational Linguistics.

Zhiting Hu, Haoran Shi, Bowen Tan, Wentao Wang, Zichao Yang, Tiancheng Zhao, Junxian He, Lianhui Qin, Di Wang, Xuezhe Ma, Zhengzhong Liu, Xiaodan Liang, Wanrong Zhu, Devendra Singh Sachan, and Eric P. Xing. 2019. Texar: A modularized, versatile, and extensible toolkit for text generation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28 - August 2, 2019, Volume 3: System Demonstrations*, pages 159–164. Association for Computational Linguistics.

Yadi Lao, Jun Xu, Sheng Gao, Jun Guo, and Jirong Wen. 2019. Name entity recognition with policy-value networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 1245–1248. ACM.

Shen Li, Zhe Zhao, Renfen Hu, Wensi Li, Tao Liu, and Xiaoyong Du. 2018. Analogical reasoning on chinese morphological and semantic relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2:*

Short Papers, pages 138–143. Association for Computational Linguistics.

Zachary C. Lipton, Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng. 2018. Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems. 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), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5237–5244. AAAI Press.

Shuman Liu, Hongshen Chen, Zhaochun Ren, Yang Feng, Qun Liu, and Dawei Yin. 2018. Knowledge diffusion for neural dialogue generation. 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 1489–1498. 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 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL.

Jinghui Qin, Zheng Ye, Jianheng Tang, and Xiaodan Liang. 2020. Dynamic knowledge routing network for target-guided open-domain conversation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8657–8664. AAAI Press.

Lianhui Qin, Michel Galley, Chris Brockett, Xiaodong Liu, Xiang Gao, Bill Dolan, Yejin Choi, and Jianfeng Gao. 2019. Conversing by reading: Contentful neural conversation with on-demand machine reading. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5427–5436. 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.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy P. Lillicrap, and David Silver. 2019. Mastering atari, go, chess and shogi by planning with a learned model. *CoRR*, abs/1911.08265.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 3295–3301. AAAI Press.

Karin Sevegnani, David M. Howcroft, Ioannis Konstas, and Verena Rieser. 2021. Otters: One-turn topic transitions for open-domain dialogue. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 2492–2504. Association for Computational Linguistics.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of go with deep neural networks and tree search. *Nat.*, 529(7587):484–489.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. 2017. Mastering the game of go without human knowledge. *Nat.*, 550(7676):354–359.

Jianheng Tang, Tiancheng Zhao, Chenyan Xiong, Xiaodan Liang, Eric P. Xing, and Zhiting Hu. 2019. Target-guided open-domain conversation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5624–5634. Association for Computational Linguistics.

Sihan Wang, Kaijie Zhou, Kunfeng Lai, and Jianping Shen. 2020. Task-completion dialogue policy learning via monte carlo tree search with dueling network. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 3461–3471. Association for Computational Linguistics.

Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. 2017. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the*

55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers, pages 496–505. Association for Computational Linguistics.

Sanghyun Yi, Rahul Goel, Chandra Khatri, Alessandra Cervone, Tagyoung Chung, Behnam Hedayatnia, Anu Venkatesh, Raefer Gabriel, and Dilek Hakkani-Tür. 2019. Towards coherent and engaging spoken dialog response generation using automatic conversation evaluators. In *Proceedings of the 12th International Conference on Natural Language Generation, INLG 2019, Tokyo, Japan, October 29 - November 1, 2019*, pages 65–75. Association for Computational Linguistics.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? 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 2204–2213. Association for Computational Linguistics.

Peixiang Zhong, Yong Liu, Hao Wang, and Chunyan Miao. 2020. Keyword-guided neural conversational model. *CoRR*, abs/2012.08383.

Xiangyang Zhou, Daxiang Dong, Hua Wu, Shiqi Zhao, Dianhai Yu, Hao Tian, Xuan Liu, and Rui Yan. 2016. Multi-view response selection for human-computer conversation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 372–381. The Association for Computational Linguistics.

Xiangyang Zhou, Lu Li, Daxiang Dong, Yi Liu, Ying Chen, Wayne Xin Zhao, Dianhai Yu, and Hua Wu. 2018. Multi-turn response selection for chatbots with deep attention matching network. 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 1118–1127. Association for Computational Linguistics.

Resolving Prepositional Phrase Attachment Ambiguities with Contextualized Word Embeddings

Adwait Ratnaparkhi
University of California Santa Cruz
U.S.A.
adratnap@ucsc.edu

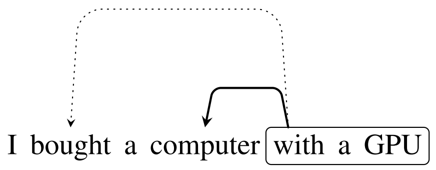
Atul Kumar
Roku
U.S.A.
kumatul06@gmail.com

Abstract

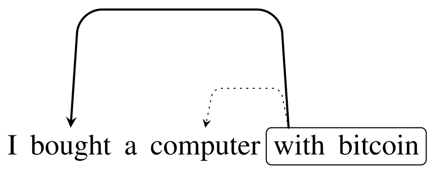
This paper applies contextualized word embedding models to a long-standing problem in the natural language parsing community, namely prepositional phrase attachment. Following past formulations of this problem, we use data sets in which the attachment decision is both a binary-valued choice as well as a multi-valued choice. We present a deep learning architecture that fine-tunes the output of a contextualized word embedding model for the purpose of predicting attachment decisions. We present experiments on two commonly used datasets that outperform the previous best results, using only the original training data and the unannotated full sentence context.

1 Introduction

Prepositional phrase (PP) attachment is a subproblem of natural language parsing in which the objective is to determine the likely attachment site of the preposition. The attachment site should correspond to the preferred semantic interpretation of the sentence.



In the above example, the preposition *with* could attach to *bought* or *computer*. The likely interpretation is that *GPU* is a sub-component of *computer*, which implies an attachment to *computer*.



In this example, the likely interpretation is that *bitcoin* is a payment method, which implies an attachment to *bought*.

PP attachment ambiguities are difficult to resolve because the candidate attachment sites look equally plausible from the perspective of natural language syntax. Deciding the best attachment site for a preposition often requires a semantic interpretation of the words in the sentence.

2 Previous work

Early work on this task uses relationships between *head words* of the phrases involved in the attachment decision. Hindle and Rooth (1993) predict attachments using co-occurrence statistics between the preposition and the candidate heads, drawn from an automatically built corpus of partial parses. Ratnaparkhi et al. (1994); Brill and Resnik (1994); Collins and Brooks (1995) use a wider variety of machine learning techniques to learn the attachment decision from annotated tuples of head words.

Later work uses a variety of external data sources to help with the attachment decision. E.g., Stetina and Nagao (1997) use features from WordNet (Miller, 1995) while Olteanu and Moldovan (2005) use features from FrameNet (Baker et al., 1998) and co-occurrence statistics drawn from the World Wide Web.

More recent work uses word embeddings and neural models. Belinkov et al. (2014) explore a number of neural composition architectures to combine the embeddings from the head words, while Dasigi et al. (2017) use the WordNet ontology to create context-sensitive word embeddings. Yu et al. (2016) use a scoring function on low-rank tensors, created from a variety of PP attachment features, while Madhyastha et al. (2017) use the tensor products of the word vectors in a multi-linear model. Recently, Do and Rehbein (2020) present German language PP attachment experiments using a neural scoring model with a biaffine transformation.

Field	Example
Verb h_1	made
Noun h_2	paper
Preposition p	for
Child noun $n2$	filters
Label	N

Table 1: Example in RRR data set. h_1 and h_2 are the verb and noun candidate attachment sites for p, respectively.

Field	Example
Candidate heads $h_1 \ldots h_n$	made trip compete
Preposition p	for
Child noun $n2$	slots
Full sentence	eventually , about 250 made the trip to florida to compete for the available slots .
Annotated label	3

Table 2: Example in BLBG data set with joined sentence. h_3, or "compete", is the annotated attachment site for the preposition.

3 Data

In this work, we report results on the English-language data sets from Ratnaparkhi et al. (1994) and Belinkov et al. (2014), henceforth referred to as the RRR and BLBG data sets, respectively. Both datasets were extracted from the Penn Treebank (Marcus et al., 1993).

Each example in the RRR data set[1] consists of a tuple $(h_1, h_2, p, n2, L)$, where h_1 is the candidate verb head, h_2 is the candidate noun head, p is the preposition, $n2$ is the noun head child of the preposition, and $L \in \{N, V\}$ is the label that indicates a noun or verb attachment. We map $\{N, V\}$ to $\{2, 1\}$ for consistency with the BLBG format, described below.

Each example in the BLBG data set[2] consists of a tuple $(h_1 \ldots h_n, p, n2, L)$, in which $h_1 \ldots h_n$ are a list of candidate noun and verb heads, p is the preposition, $n2$ is the noun head child of the preposition, and $L \in \{1 \ldots n\}$ denotes the index of the correct attachment in the list of heads. Compared to the RRR dataset, the BLBG dataset is a better approximation of the attachment decision faced in a full parsing task since it allows more than two possible attachment sites.

[1]https://github.com/adwaitratnaparkhi/ppa_transformer
[2]http: //groups.csail.mit.edu/rbg/code/pp

	Training	Development	Test
RRR	20801	4039	3097
BLBG	35359	n/a	1951

Table 3: Data set sizes of the RRR and BLBG data sets

In order to enable experiments in which the full sentence context is used as input, the original head word tuples were joined with the full sentences from which they were extracted. Starting from data generation scripts provided to us by the author[3] of the BLBG data set, we joined each example in the original BLBG data set with its full sentence context from the Penn Treebank Version 3. The sentences were lower cased to match the convention of the BLBG data. As we could not obtain the matching version of the Penn Treebank for the RRR dataset[4], the full sentence experiments were only conducted on the BLBG data set.

Table 3 shows the size of the data sets. Tables 1 and 2 contain example training instances.

4 Our method

In recent years, contextualized word embeddings (CWE), particularly those built using the Transformer (Vaswani et al., 2017) architecture, have accelerated progress in many NLP tasks. Past work on NLP tasks has typically followed a transfer learning strategy, in which a large pre-trained model is fine-tuned on a small annotated training set with task-specific labels. We apply this strategy using pre-trained BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019) models on data specific to the prepositional phrase attachment task, with the objective of improving task accuracy.

Figure 1 introduces a deep learning architecture that fine-tunes the output of a CWE module in order to predict attachments. We henceforth refer to it as the Fine Tuning for Headword Attachment (FTHA) model[5]. The full sentence or phrase is first passed through the CWE module, which yields a vector of token embeddings, shown in layer (1). In cases where the original word has been split into multiple tokens by the CWE tokenizer, we follow the convention in (Devlin et al., 2018) and use the

[3]Many thanks to Yonatan Belinkov for the data generation code.
[4]The RRR dataset cannot be joined with Penn Treebank Version 3 due to missing data.
[5]The FTHA implementation can be found at https://github.com/adwaitratnaparkhi/ppa_transformer

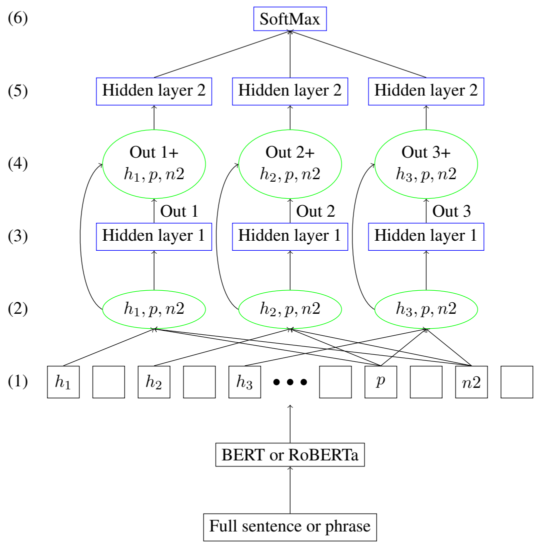


Figure 1: The Fine Tuning for Headword Attachment (FTHA) model. The diagram shows only 3 candidate head word attachment sites for the preposition, but the network can operate on an arbitrary number of candidate head words.

embeddings of the first sub-token. Next, a mask operation extracts a vector of n embedding triples $(h_1, p, n2) \ldots (h_n, p, n2)$, shown in layer (2). Each triple represents a possible attachment. The embedding triples are passed to a hidden layer (3), and then concatenated with the hidden layer output using a "skip" connection, shown in layer (4). The result (4) is then passed to a second hidden layer (5). Finally, a softmax layer (6) returns a score for each attachment. The hidden layers in (3) and (5) share their parameters across their respective layer.

Each input example contains the information shown in Tables 1 and 2, together with a token sequence, and token indices of the head words. Depending on the experiment configuration, the token sequence is either the full sentence from which the example was drawn, or a phrase synthesized by concatenating the words in $(h_1 \ldots h_n, p, n2)$.

5 Experiments

The FTHA model is trained and evaluated on both the RRR and BLBG data sets. Furthermore, we compare with past results on both data sets, as well as against an existing implementation[6] of a multiple choice fine-tuning architecture in the HuggingFace (HF) code library (Wolf et al., 2019).

The HF multiple choice implementation was designed for the SWAG (Zellers et al., 2018) task, and is described in Devlin et al. (2018). We configure the implementation to predict up to 8 choices, where each choice is a phrase formed from the head word triple $(h_i, p, n2)$, or a dummy label. For context input, we also give it either a full sentence or a phrase synthesized from all of the head words, depending on the experiment configuration. For our experiments, we configure it to use RoBERTa.

The HF implementation encodes the text of the choice (i.e., the head word triples) and any additional input, "pools" the embeddings, and then passes them to linear and softmax layers. In contrast, the FTHA model retains the head word embeddings that were computed in the context of the full sentence or phrase. This representation allows the FTHA model to work with a higher granularity of information from the input, compared to the HF implementation.

All experiments are measured using accuracy of the label classification. The allowable labels for each data set and their distributions are shown in Table 4.

Data set	Label	Frequency	Label	Frequency
RRR	N	10865	V	9936
BLBG	1	1585	5	4957
	2	7961	6	1478
	3	10113	7	225
	4	9022	8	18

Table 4: Allowable labels and their distribution in the RRR and BLBG training sets.

Table 5 shows results on the RRR dataset using the FTHA model with both BERT and RoBERTa. Only the head word tuples are used as input. The best result, FTHA with RoBERTa, is 0.7% higher than the one reported in Stetina and Nagao (1997), which is the previously best known result for the RRR dataset.

Table 6 shows results on the BLBG dataset with multiple experiment configurations. The best result in the `Head words only` configuration, FTHA with RoBERTa, outperforms the previously best result from Yu et al. (2016) by 1.7%. The best result in the `Full sentence` configuration, again FTHA with RoBERTa, outperforms that result by 4.1%. Table 7 shows an example where having the full sentence context helps.

The higher granularity representation of the FTHA model (with RoBERTa) gives slight accu-

[6]https://github.com/huggingface/transformers/tree/master/examples/multiple-choice

Paper	Dev	Test
(Stetina and Nagao, 1997)	n/a	88.1%
HF multiple choice	89.1%	88.4%
FTHA (RoBERTa)	89.3%	**88.8%**
FTHA (BERT)	88.8%	87.9%

Table 5: Results on the RRR data set.

Paper/Configuration and Model	Accuracy
(Belinkov et al., 2014)	
HPCD-full	88.7%
HPCD-full with parser	90.1%
(Dasigi et al., 2017)	
OntoLSTM-PP	89.7%
OntoLSTM-PP with parser	90.11%
(Yu et al., 2016)	
LRFR1-TUCKER & LRFR2-CP	90.3%
`Head words only` configuration	
FTHA (RoBERTa)	**92.0%**
FTHA (BERT)	91.1%
HF multiple choice	91.1%
`Full sentence` configuration	
FTHA (RoBERTa)	**94.4%**
FTHA (BERT)	93.2%
HF multiple choice	94.2%

Table 6: Results on the BLBG data set

racy gains compared to the HF baselines, for both the RRR and BLBG experiments. However, the differences are not statistically significant according to McNemar's test at $\alpha = 0.05$ significance level.

All RoBERTa and BERT experiments use the `roberta-base` and `bert-base-uncased` models, respectively. All models have at most 126M parameters, and training times were at most 1 hour on a UNIX server with an NVIDIA Titan RTX GPU with 24Gb RAM.

5.1 Hyperparameters

In our experiments, the hidden layers 1 and 2 in Figure 1 have a size of $3N$ and $4N$, respectively, where $N = 768$. During training, we use 3 epochs, a batch size of 16, the `AdamW` optimizer, a warmup step of 500, a weight decay of 0.01, and a learning rate of 10^{-3}.

6 Discussion

In past work, data sparsity has been a major challenge for corpus-based approaches to prepositional phrase attachment. To remedy data sparsity, researchers have incorporated features from external resources like WordNet and FrameNet, with the idea that semantic features on words will have far less than sparsity than words themselves.

Field	Example
Candidate heads $h_1 \dots h_n$	plan impose freeze
Preposition p	on
Child noun $n2$	fees
Full sentence	the plan would impose a brief freeze on physician fees next year .
Annotated label	2

Table 7: Test example where the `Head words only` configuration predicts incorrectly and the `Full sentence` configuration predicts correctly, for the FTHA (RoBERTa) model. h_2, or "impose", is the annotated attachment site for the preposition.

In our work, the CWE, built from a huge amount of unsupervised data, seem to compensate for the sparsity in the relatively small training sets. Our results on both the RRR and BLBG sets exceed those reported in past work, but without using WordNet, FrameNet, or any other external resources. All other results compared in Tables 6 and 5 use resources external to the training set.

We do not compare our experimental results with those in Do and Rehbein (2020) and Olteanu and Moldovan (2005) due to differences in test data.

The approach of Do and Rehbein (2020), conducted on German-language PP attachment data, resembles our work in that it uses CWE in a neural model. It also presents a scoring function for $(h_i, p, n2)$ triples, using a biaffine transformation of word representations derived from a bidirectional LSTM. It differs from our approach in that it considers all words in the sentence as potential attachment sites. And unlike our work, it incorporates additional information like part-of-speech tags, topological field tags, and auxiliary distributions computed from a large newspaper corpus.

The approach of Olteanu and Moldovan (2005) uses support vector machines and requires rich features that cannot be derived from the RRR dataset. Therefore it creates a new and larger data set with complex features from syntax trees, FrameNet, and co-occurrence statistics derived from an internet search engine. In contrast, our work focuses on only the CWE and excludes external resources.

Both the FTHA and HF experiments show benefit from using the full sentence context. For FTHA, the embeddings computed in the context of the whole sentence are likely more accurate than those computed from the head word phrase. For the HF implementation, the pooled embedding of the sentence carries enough information to boost the accuracy over just using the head words. Notably, in both cases, the sentence information is not explicitly annotated with any syntactic information, yet gives a sizable boost (+ 2% to 3%) in PP attachment accuracy.

Our work studies the PP attachment problem in isolation, and does not compare against the attachment decisions of a full parser. Other work (Belinkov et al., 2014; Dasigi et al., 2017) shows that PP attachment is still a problematic area for full parsers, and that an independently trained PP attachment model can improve the decisions of a full parser.

7 Conclusion

We present deep learning experiments for the prepositional phrase attachment task that exceed the accuracy of all previously published results on two widely used data sets. The results in our paper are 0.7% and 4.1% higher in absolute percentage points over the best previously published results on the RRR and BLBG data sets, respectively. We present a novel fine-tuning architecture that uses a higher granularity of information from the input, compared to a baseline implementation from HuggingFace. All our results were obtained without using external semantic data sources like WordNet or FrameNet. Lastly, we observe a big accuracy gain when the model is given the full sentence context vs. only the head words, despite having no syntactic annotation in the full sentence context.

References

C. Baker, C. Fillmore, and J. Lowe. 1998. The Berkeley FrameNet Project. In *COLING-ACL*.

Yonatan Belinkov, Tao Lei, Regina Barzilay, and Amir Globerson. 2014. Exploring compositional architectures and word vector representations for prepositional phrase attachment. *Transactions of the Association for Computational Linguistics*, 2:561–572.

Eric Brill and Philip Resnik. 1994. A rule-based approach to prepositional phrase attachment disambiguation. In *COLING 1994*.

Michael Collins and James Brooks. 1995. Prepositional phrase attachment through a backed-off model. In *Proceedings of the Third Workshop on Very Large Corpora*.

Pradeep Dasigi, Waleed Ammar, Chris Dyer, and Eduard Hovy. 2017. Ontology-aware token embeddings for prepositional phrase attachment.

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.

Bich-Ngoc Do and Ines Rehbein. 2020. Parsers know best: German PP attachment revisited. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2049–2061.

Donald Hindle and Mats Rooth. 1993. Structural ambiguity and lexical relations. *Computational linguistics*, 19(1):103–120.

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*.

Pranava Swaroop Madhyastha, Xavier Carreras, and Ariadna Quattoni. 2017. Prepositional phrase attachment over word embedding products. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 32–43.

M. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Comput. Linguistics*, 19:313–330.

G. Miller. 1995. Wordnet: a lexical database for english. *Commun. ACM*, 38:39–41.

Marian Olteanu and Dan Moldovan. 2005. PP-attachment disambiguation using large context. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 273–280.

Adwait Ratnaparkhi, Jeff Reynar, and Salim Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Jiri Stetina and Makoto Nagao. 1997. Corpus based PP attachment ambiguity resolution with a semantic dictionary. In *Fifth Workshop on Very Large Corpora*.

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.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Mo Yu, Mark Dredze, Raman Arora, and Matthew R Gormley. 2016. Embedding lexical features via low-rank tensors. In *Proceedings of NAACL-HLT*, pages 1019–1029.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. *CoRR*, abs/1808.05326.

Multi-Source Cross-Lingual Constituency Parsing

Hour Kaing[†‡], **Chenchen Ding**[†], **Katsuhito Sudoh**[‡], **Masao Utiyama**[†], **Eiichiro Sumita**[†], **Satoshi Nakamura**[‡]
[†]National Institute of Information and Communications Technology,
3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0289, Japan
[‡]Nara Institute of Science and Technology,
8916-5 Takayama-cho, Ikoma, Nara 630-0192, Japan

Abstract

Pretrained multilingual language models have become a key part of cross-lingual transfer for many natural language processing tasks, even those without bilingual information. This work further investigates the cross-lingual transfer ability of these models for constituency parsing and focuses on multi-source transfer. Addressing structure and label set diversity problems, we propose the integration of typological features into the parsing model and treebank normalization. We trained the model on eight languages with diverse structures and use transfer parsing for an additional six low-resource languages. The experimental results show that the treebank normalization is essential for cross-lingual transfer performance and the typological features introduce further improvement. As a result, our approach improves the baseline F1 of multi-source transfer by 5 on average.

1 Introduction

Recent pretrained multilingual language models have become a key step in cross-lingual transfer for many natural language processing tasks such as name entity recognition, part-of-speech tagging, natural language inference, and dependency parsing (Wu and Dredze, 2019). These models are desirable in research on cross-lingual transfer because bilingual information is not required.

Cross-lingual transfer is when a trained model for a source language is applied to a target (unseen) language. There are two transfer scenarios, single-source and multi-source transfer. For single-source transfer, each time, the model is trained on only one source language. In this scenario, multiple models are available for cross-lingual transfer in practice. Additional model selection is necessary for single-source transfer because cross-lingual transfer relies on language isomorphism. For multi-source transfer, to leverage all existing resources, treebanks of multiple languages are combined to train a multilingual parser that can be later used for any unseen language. In this work, we study the multi-source transfer for sophisticated structure prediction, i.e., constituency parsing. Our work will serve as a benchmark for cross-lingual constituency parsing using pretrained multilingual language model.

For constituency parsing, training a multilingual parser has two main issues that must be considered. First, the source languages can produce diverse word orders—for instance, different *subject-verb-object* or *noun-adjective* orders. These language properties can be simply identified using existing typology databases, e.g., The World Atlas of Language Structures (WALS) or Syntactic Structures of the World's Languages (SSWL). It is intuitive that these language properties can be used to guide a multilingual parser to share corresponding model parameters among similar languages (Naseem et al., 2012; Ammar et al., 2016; Scholivet et al., 2019; Üstün et al., 2020). For cross-lingual transfer, the typological features could hurt performance (Ammar et al., 2016), and an effective integration technique is required (Üstün et al., 2020). Inspired by this, we investigate the usefulness of typological features for cross-lingual constituency parsing and propose a training strategy to generalize the cross-lingual capability of the model using smooth sampling and random dropout.

The second issue is that even though constituency structure is universal, the design of a label set is language specific. For dependency structures, this problem has inspired the creation of the Universal Dependency project (Nivre et al., 2016). The syntactic label sets of constituency structure vary across languages—for instance, very few labels are shared and even labels for the same syntactic category may be different across languages. This increases the complexity of multi-source transfer. Therefore, we propose normalization of the con-

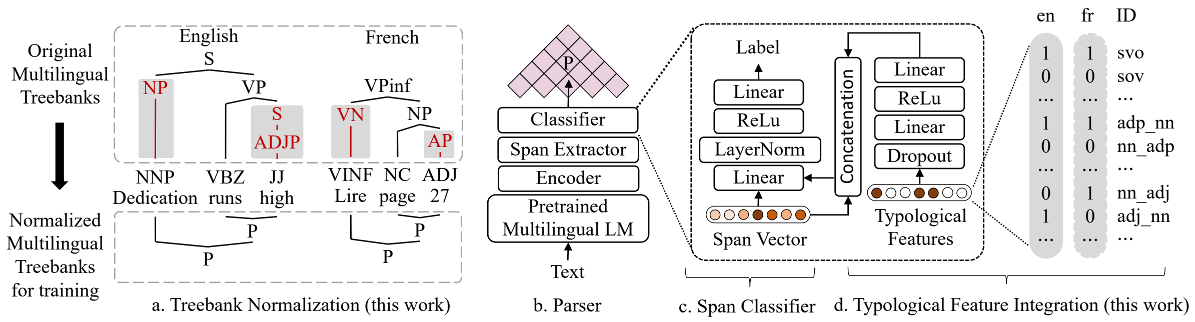


Figure 1: Overall architecture of our parser. The multilingual treebanks are normalized (a) before training the parser (b). A span classifier (c) is also integrated with a feature extractor (d) for binary typological vectors, as shown in the right-most example.

stituency treebanks to universalize the multilingual parsing model.

The contributions of this paper are summarized as follows: 1) typological feature integration for model generalization on unseen languages (Section 4.1), and 2) treebank normalization is proposed to reduce the complexity of cross-lingual structural prediction (Section 4.2).

2 Related Work

In multi-source transfer, task-specific knowledge of multiple source languages is combined and jointly transferred to an unseen or zero-shot language. This combination can be categorized according to three levels (Das and Sarkar, 2020), that is, the level of treebanks (McDonald et al., 2011; Ammar et al., 2016; Scholivet et al., 2019; Üstün et al., 2020), model parameters (Cohen et al., 2011; Søgaard and Wulff, 2012), or parse outputs (Rosa and Žabokrtský, 2015; Agić, 2017). This work focuses on treebank level, that is, treebank concatenation and, unlike previous studies, we study a more sophisticated structure, constituency treebanks, which simultaneously contain diverse syntactic labels across multiple source languages.

Typological features are a valuable resource for multi-source transfer where source languages have diverse structures, and they have been used specifically for sharing the parameters of non-neural (Naseem et al., 2012; Täckström et al., 2013; Zhang and Barzilay, 2015) and neural (Ammar et al., 2016; Scholivet et al., 2019; Üstün et al., 2020) models. Following the same motivation, we also investigate the usefulness of typological features for a multilingual constituent parser and propose a training strategy that generalizes the model for zero-shot languages. Specifically, we integrate typological features into the self-attentive constituency parser (Kitaev and Klein, 2018).

Our work is similar to that of Kitaev et al. (2019) who investigated the multilingualism of the self-attentive constituency parser (Kitaev and Klein, 2018) using the pretrained multilingual language model. However, our work differs from theirs such that we focus on zero-shot performance. In addition, we propose to normalize the concatenated treebanks and integrate typological features for better zero-shot performance. We also extend the sampling technique that Kitaev et al. (2019) use by constraining the minimum size of each treebank.

3 The Self-Attentive Parser

The basis of our model (Fig. 1b) follows the self-attention based encoder–decoder architecture of Kitaev and Klein (2018). Specifically, the encoder consists of word embedding and self-attention layers to produce the contextual presentation for each word. At the decoder side, all possible spans are extracted and each span (i, j) is represented by a hidden vector $v_{i,j}$ that is constructed by subtracting the representations associated with the start and end of the span. Then, each span (i, j) is assigned a labeling score $s(i, j, \cdot)$ by an MLP span classifier as

$$s(i, j, \cdot) = W_2 g(f(W_1 v_{i,j} + c_1)) + c_2, \quad (1)$$

where W_* and c_* are the weight and bias, respectively; f and g are the layer normalization and ReLU (”Re”ctified ”L”inear ”U”nit) activation function, respectively, as shown in Figure 1c. For each sentence, the constituency structure T is represented by a set of labeled spans $\{(i_t, j_t, l_t) : t = 1, \dots, |T|\}$ where l is a label. Therefore, the score of T is

$$s(T) = \sum_{(i,j,l)\in T} s(i, j, l). \quad (2)$$

At test time, the optimal structure can be obtained using a CKY-style inference algorithm. For training, the model is optimized using a max-margin objective function, the details of which can be found in Kitaev and Klein (2018). In addition, the parser's hyperparameters are unchanged from Kitaev and Klein (2018).

To perform cross-lingual parsing, an external pre-trained multilingual language model must be used and simply take the place of the word embedding layer. Because the model is trained on sub-words, only the last sub-word unit of the corresponding token is used to represent a word. In this experiment, we use a recent multilingual language model, i.e. XLM-RoBERTa-Large (Conneau et al., 2020).

4 Proposed Methods

4.1 Typological Feature Integration

A typology database is a valuable resource that represents various aspects of languages. Recent `lang2vec` (Littell et al., 2017) provides an interface to represent languages as binary vectors of typological features. Inspired by the recent work of Üstün et al. (2020), we also integrate typological features f (TF) into our model to guide the multilingual model's sharing of the structural knowledge among similar languages, as Figure 1d shows. We use simple feature concatenation to integrate typological features into the span classifier. Like Üstün et al. (2020), we embed binary typological vectors using two linear layers and a ReLU activation function g, and further apply random dropout over the binary typological vectors as

$$f' = M_2 g(M_1 dropout(f) + z_1) + z_2. \quad (3)$$

We then concatenate f' with each span vector, $v_{i,j}$, which modifies Equation 1 as

$$s(i, j, \cdot) = W_2 g(f(W_1[v_{i,j}, f'] + c_1)) + c_2. \quad (4)$$

Dropout is applied directly to the binary features because, during training, typological features only vary with respect to the number of source languages, and each feature is only helpful in the context of other features, which is known as co-adaptation (Hinton et al., 2012). Therefore, for a zero-shot language, without dropout, the model would not be able to extract individual features in a new feature context, which can be prevented using simple random dropout (Hinton et al., 2012). Like Hinton et al. (2012), we drop 50% of the features during training.

The number of multilingual treebanks commonly differs, and high-resource languages tend to be over-represented during training. Similar to the exponential smoothing in Kitaev et al. (2019), at each epoch, we sample d^a examples from each language, where d is the size of each language treebank and a is a hyperparameter. Unlike Kitaev et al. (2019), we use $a = 0.95$ because the size of each treebank is not as large as the unlabeled corpora. We also constrain the smoothed number of examples as $d^a > m$, where m is the smallest treebank size in the source-language pool. We call this approach "smooth sampling."

For the typological features, we combine the syntax features of WALS (Dryer and Haspelmath, 2013) or SSWL (Collins and Kayne, 2011)[1]. We only select the relevant features such as 81A, 82A, 83A, 85A, 87A, 88A, 89A, 90A, 144A, and other unknown ID features such as *subject_b/a_object*[2], *possessor_b/a_noun*, *degree_word_b/a_adjective*, and *subordinator_word_b/a_clause*. In addition, we exclude the morphological features, which contain the word *prefix* or *suffix*, and the missing features of any source language. For zero-shot languages, the missing features are set to zero. After that, we further automatically remove unnecessary features that are repeated for all source languages. Like Ustun et al. (Üstün et al., 2020), we set the hidden and output layer of our TF to 10 and 32, respectively.

4.2 Treebank Normalization

Another obvious issue of constituency treebanks is the difference in their syntactic labels. We observed that high-resource languages tend to have more diverse labels, whereas low-resource languages use a much smaller label set; for instance, Myanmar and Khmer have five and six labels, respectively, whereas English has 26. Moreover, label symbols for each treebank are very language specific; for example, French and English, which have large label sets, only share two labels.

Therefore, we propose treebank normalization (TN) as the preprocessing step in our approach. Specifically, we first remove any non-terminal span that has length or number of children less than two. In other words, they are any span $(i, j) \in T$ where $j - i < 2$. After that, we mask the labels of all the remaining non-terminal spans with an unified symbol, e.g., "P" as in the example in Figure 1a.

[1]These features can be obtained using `lang2vec` by passing a `syntax_wals+syntax_sswl.` argument.

[2]b/a denote "before or after".

Code	Language	Train	Valid	Test
`de`	German	40,472	5,000	5,000
`en`	English	39,832	1,700	2,416
`ko`	Korean	23,010	2,066	2,287
`my`	Myanmar	18,088	1,000	1,018
`zh`	Chinese	17,544	352	348
`ja`	Japanese	17,204	953	931
`ar`	Arabic	15,762	1,985	1,959
`fr`	French	14,759	1,235	2,541
`km`	Khmer	8,788	510	654
`hu`	Hungarian	8,146	1,051	1,009
`eu`	Basque	7,577	948	946
`pl`	Polish	6,578	821	822
`sv`	Swedish	5,000	494	666
`he`	Hebrew	5,000	500	716

Table 1: Data statistics. The numbers refer to numbers of sentences where upper languages are high-resource languages and lower for low-resource languages.

Lang.	S_{best}	S_{dist}	M_{base}	TN_{ours}	$+ TF_{ours}$
km	70.0	70.0	55.5	69.0	**71.8**
hg	64.7	31.2	68.6	73.9	**74.7**
eu	33.2	27.2	27.3	34.7	**35.8**
pl	**72.8**	**72.8**	65.6	67.9	68.3
sv	**74.8**	**74.8**	67.8	73.1	73.8
he	77.1	71.3	80.5	81.6	**82.2**
avg	64.5	55.5	61.9	66.2	**67.0**

Table 2: Main unlabeled F1 results. The best F1 for each row is highlighted in bold text.

As a result, our label classifier is simplified to only detect the span as a span or non-span.

5 Experiments

5.1 Setups

The evaluation was performed on 14 languages: English from the Penn Treebank (Marcus et al., 1993); Chinese from the Chinese Penn Treebank 5.1 (Xue et al., 2005); Japanese, Khmer, and Myanmar (`my`) from the Asian Language Treebank (Riza et al., 2016); and Arabic, Basque, French, German, Hebrew, Hungarian, Korean, Polish, and Swedish from the SPMRL 2013 shared task (Seddah et al., 2014). The standard splits of each treebank were applied to prepare the training, validation, and test datasets.

We grouped the languages into high- and low-resource (zero-shot) languages based on their amount of data; those with fewer than 10k samples were treated as low-resource languages (Khmer, Hungarian, Basque, Polish, Swedish, and Hebrew). We trained a multilingual model on the high-resource languages and evaluated the cross-lingual parsing on the low-resource languages.

Note that Khmer and Myanmar scripts have no word boundaries, so we simply use their gold segmented long token[3] for this experiment. We observe that XLM-RoBERTa-Large's tokenizer produces reasonable sub-words for Khmer and Myanmar's long tokens, even when the tokenizer was trained using SentencePiece for these two languages. Table 1 presents detailed data statistics for each language.

For comparison, we trained two baselines, single- and multi-source models. For the single-source model, we trained parser for each high-resource language and then selected the best model based on its parsing accuracy on the oracle test set (S_{best}) of the low-resource language or used the precomputed syntactic distance (Littell et al., 2017) (S_{dist}). For the same-value syntactic distance, we further weighted each source language based on the size of its corresponding training data. For the multi-source baseline, a multilingual parser (M_{base}) was trained on concatenated treebanks without treebank normalization or typological features.

Because the label sets of each treebank differ and calculating the accuracy of label prediction was difficult, we calculated the unlabeled F1 measure to evaluate cross-lingual performance. All following F1 values refer to the unlabeled F1 for simplicity. We also removed unnecessary spans such as sentence-level and length-of-one spans.

5.2 Results

As shown in Table 2, the performance of single-source transfer was very high, especially when the best source language can be accurately detected. Unfortunately, the precomputed syntactic distance is not enough to choose the best source language; in the results, it failed in three out of six cases. The alternative to source selection is to train a multilingual parser. Interestingly, even the straightforward treebank concatenation M_{base} has a competitive performance when compared with single-source transfer. The results further show that treebank normalization is essential when training a multilingual

[3]Khmer and Myanmar written scripts can be segmented into morphemes (short tokens) or at compound level (long tokens) (Ding et al., 2018)

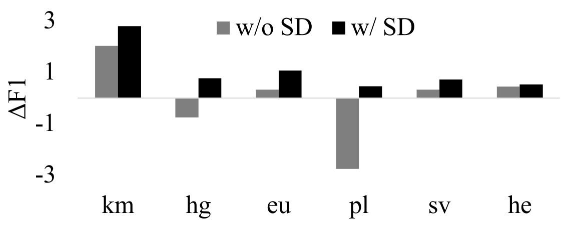


Figure 2: Improvements in the F1 of TN+TF over TN model with or without SD. SD refers to **S**mooth sampling and random **D**ropout.

constituency parser for zero-shot languages, where the improvement over $\mathrm{M}_{\mathrm{base}}$ is 4.3 in average F1. This result suggests that reducing the complexity of the structure improves cross-lingual performance.

In addition to treebank normalization, our integration of typological features constantly improves cross-lingual performance. An analysis of Figure 2 further shows that the straightforward integration of typological feature yields smaller improvements or hurts the performance for some zero-shot languages, indicating the effectiveness of our smooth sampling and dropout, which generalize the typology-guided cross-lingual parser for zero-shot languages. We additionally observe that the combination of both smooth sampling and dropout is the best configuration for the cross-lingual parsing.

6 Conclusion

We demonstrated the strong ability of recent pre-trained multilingual language models for cross-lingual constituency parsing. This result will serve as a new benchmark for future cross-lingual constituency parsing. Moreover, we found that our treebank normalization is crucial when training multilingual treebanks with diverse label sets. In addition, our typological feature integration with dropout and smooth sampling generalizes and improves the model for zero-shot languages. Because we integrated typological features into the span classifier using a simple concatenation approach, more advanced techniques—for instance, a parameter generator (Üstün et al., 2020)—with our dropout and smooth sampling could be studied in the future.

Additionally, our parser could make the applications that leverage structures possible for a wide range of languages without additional treebanks. For example, pseudo constituency structures that our parser generate could be used to apply the recurrent neural network grammar (Dyer et al., 2016; Kim et al., 2019) or the syntax-based neural machine translation (Ma et al., 2019) for many non-English languages. However, since the pseudo structures could be noisy or irrelevant to the model, selective or soft integration techniques should be considered (Chakrabarty et al., 2020).

References

Željko Agić. 2017. Cross-lingual parser selection for low-resource languages. In *Proc. of UDW*, pages 1–10.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.

Abhisek Chakrabarty, Raj Dabre, Chenchen Ding, Masao Utiyama, and Eiichiro Sumita. 2020. Improving low-resource nmt through relevance based linguistic features incorporation. In *Proc. of COLING*, pages 4263–4274.

Shay B Cohen, Dipanjan Das, and Noah A Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proc. of EMNLP*, pages 50–61.

Chris Collins and Richard Kayne. 2011. Syntactic structures of the world's languages.

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 *Proc. of ACL*, pages 8440–8451.

Ayan Das and Sudeshna Sarkar. 2020. A survey of the model transfer approaches to cross-lingual dependency parsing. *ACM Trans. Asian Low-Resour. Lang. Info. Process.*, 19(5):1–60.

Chenchen Ding, Masao Utiyama, and Eiichiro Sumita. 2018. Nova: A feasible and flexible annotation system for joint tokenization and part-of-speech tagging. *ACM Trans. Asian Low-Resour. Lang. Info. Process.*, 18(2):1–18.

Matthew S. Dryer and Martin Haspelmath. 2013. The world atlas of language structures online.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. pages 199—-209.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580. [cs.NE].*

Yoon Kim, Chris Dyer, and Alexander M Rush. 2019. Compound probabilistic context-free grammars for grammar induction. In *Proc. of ACL*, pages 2369–2385.

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In *Proc. of ACL*, pages 3499–3505.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proc. of ACL*, pages 2676–2686.

Patrick Littell, David R Mortensen, Ke Lin, Katherine Kairis, Carlisle Turner, and Lori Levin. 2017. Uriel and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *Proc. of EACL*, pages 8–14.

Chunpeng Ma, Akihiro Tamura, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2019. Improving neural machine translation with neural syntactic distance. In *Proc. of NAACL*, pages 2032–2037.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

Ryan McDonald, Slav Petrov, and Keith B Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proc. of EMNLP*, pages 62–72.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proc. of ACL*, page 629–637.

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 *Proc. of LREC*, pages 1659–1666.

Hammam Riza, Michael Purwoadi, Gunarso, Teduh Uliniansyah, Aw Ai Ti, Sharifah Mahani Aljunied, Luong Chi Mai, Vu Tat Thang, Nguyen Phuong Thai, Rapid Sun, Vichet Chea, Sethserey Sam, Sopheap Seng, Khin Mar Soe, Khin Thandar Nwet, Masao Utiyama, and Chenchen Ding. 2016. Introduction of the Asian language treebank. In *Proc. of O-COCOSDA*, pages 1–6.

Rudolf Rosa and Zdeněk Žabokrtský. 2015. KLcpos3-a language similarity measure for delexicalized parser transfer. In *Proc. of ACL-IJCNLP*, pages 243–249.

Manon Scholivet, Franck Dary, Alexis Nasr, Benoit Favre, and Carlos Ramisch. 2019. Typological features for multilingual delexicalised dependency parsing. In *Proc. of NAACL*, pages 3919–3930.

Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the SPMRL 2014 shared task on parsing morphologically-rich languages. In *Proc. of SPMRL*, pages 103–109.

Anders Søgaard and Julie Wulff. 2012. An empirical etudy of non-lexical extensions to delexicalized transfer. In *Proc. of COLING*, pages 1181–1190.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proc. of NAACL*, pages 1061–1071.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. 2020. Udapter: Language adaptation for truly universal dependency parsing. In *Proc. of EMNLP*, pages 2302–2315.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. In *Proc. of EMNLP-IJCNLP)*, pages 833–844.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(2):207.

Yuan Zhang and Regina Barzilay. 2015. Hierarchical low-rank tensors for multilingual transfer parsing. In *Proc. of EMNLP*, pages 1857–1867.

Kannada Sandhi Generator for *Lopa* and *Adesha Sandhi*

Musica Supriya † **Dinesh Acharya U †** **Ashalatha Nayak †** **Arjuna S R ‡**
† Department of Computer Science & Engineering
Manipal Institute of Technology
Manipal Academy of Higher Education, Manipal, Karnataka 576104 , India
‡ Department of Philosophy
Manipal Academy of Higher Education, Manipal, Karnataka 576104 , India
{musica.supriya, dinesh.acharya, asha.nayak, arjuna.sr}@manipal.edu

Abstract

Kannada is one of the major spoken classical languages in India. It is morphologically rich and highly agglutinative in nature. One of the important grammatical aspects is the concept of *sandhi*(euphonic change). There has not been a *sandhi* generator for Kannada and this work aims at basic *sandhi* generation. In this paper, we present algorithms for *lopa* and *Adesha sandhi* using a rule-based approach. The proposed method generates the *sandhied* word and corresponding *sandhi* without any help of dictionary. This work is significant for agglutinative languages especially to Dravidian languages and can be used to enhance the vocabulary for language related tasks.

1 Introduction

Kannada is one of the Dravidian Languages spoken predominantly by people of Karnataka. The Dravidian languages are highly agglutinative and morphologically rich in nature (Shashirekha and Vanishree, 2016). Euphonic change known as *sandhi* (Kumar et al., 2009) is quite common in this language.

Sandhi occurs at the character level between two valid words of a particular language based on a set of rules. It also occurs between a root word/nominal stem and suffix as well. As Huet (2009) rightly mentions the Sanskrit grammarians' point that the *sandhi* is a mandatory action in the case of compound words in Sanskrit, this condition aptly applicable to Kannada as well.

The classification of *sandhi* can be done as internal *sandhi* and external *sandhi*. The euphonic change that occurs, between a root word and suffix, and in the case of construction of a compound word, is considered as internal *sandhi*. For instance, $hU + annu$[1] $= hUvannu$ [2] which means *the flower (as an object of a verb)*. In this case, the first word *hU* is a nominal stem and *annu* is a suffix. It undergoes euphonic change and *vakAra-Agama sandhi* occurs here. Hence, it is an internal *sandhi*. An example for another internal type is - $parama + ISvara = parameSvara$ $[supreme + Lord = supreme\ Lord]$. Here both *parama* and *ISvara* are words forming the compound word $parameSvara$. If *sandhi* occurs between the characters present in two different words, then it can be called as external *sandhi*. For example: $illi + ixxAlYeV = illixxAlYeV$ $[here + she\ is = she\ is\ here]$. Both the words - $illi$ and $ixxAlYeV$ are two different words.

Another way of classification is Kannada *sandhi* and Sanskrit *sandhi*. This can be classified based on the words present in the usage. As described by Kittel (1920), Kannada has borrowed many words from Sanskrit. It is possible to identify *sandhi* corresponding to pure Kannada words or a combination of Kannada and Sanskrit words. If both the *pUrvapaxa*(first word) and *uwwarapaxa*(second word) are chosen to be Kannada words, then, Kannada *sandhi* occurs. For instance, $nAnu + illi = nAnilli$ $[I + here = I\ am\ here]$. Both the words are Kannada words and the Sanskrit *sandhi* is not applicable here. The non-application of *guNa sandhi* (a Sanskrit *sandhi*) affirms that through the previous instance. If either or both are Sanskrit words, then Sanskrit *sandhi* occurs. For instance, $deva+ISanu = deveSanu$ $[Lord+Lord\ of = Lord\ of\ all\ Lords]$. Both the words are borrowed words and will follow *guNa sandhi* of Sanskrit and, not any Kannada *sandhi*.

In this paper, our focus is on Kannada *sandhi*. There are mainly three Kannada *sand-*

[1]Accusative case marker

[2]We have used WX notation given by Gupta et al. (2010) throughout this paper to represent Kannada words.

his: *lopa*(elision) *sandhi*, *Agama*(addition) *sandhi* and *Adesha*(substitution) *sandhi* as mentioned by Sharma (2015).

- *Lopa(elision) sandhi*: When *pUrvapaxa* is ending with a Kannada vowel , *uwwarapaxa* begins with a Kannada vowel, then euphonic union of this will cause the vowel at the *pUrvapaxa* to be eliminated and this word be combined with the *uwwarapaxa* to form the new word.
  Example: $nAnu + illi = nAnilli$. $[I + here = I\ am\ here]$
  Here the last vowel of *pUrvapaxa*, *u* is eliminated from *pUrvapaxa* and then joined to the *uwwarapaxa* to form the new word.

- *Agama(addition) Sandhi*: When *pUrvapaxa* is ending with a Kannada vowel , *uwwarapaxa* begins with a Kannada vowel, then euphonic union of this will cause the addition of *y* or *v* at the beginning of *uwwarapaxa*. This combination leads to *Agama sandhi*.
  Example: $maneV + ixu = maneVyixu$ $[house + this = this\ is\ house]$. Here the first vowel *i* of *uwwarapaxa* is prefixed with *y* and then joined to form the new word.

- *Adesha(substitution) Sandhi*: When *pUrvapaxa* is a Kannada word, *uwwarapaxa* begins with either *k* , *w* or *p* then it is replaced with *g*, *x*, *b* respectively .This combination of *pUrvapaxa* and *uwwarapaxa* leads to *Adesha sandhi*.
  Example: $malYeV{+}kAla = malYeVgAla$ $[rain + season = rainy\ season]$. Here *uwwarapaxa* begins with *k*, in the euphonic change, it is replaced with *g* to form the new word.

The Kannada *sandhi* can be further sub-categorized as *svara*(vowel) *sandhi* and *vyaFjana*(consonant) *sandhi*. The *svara sandhi* has vowel at the end of the *pUrvapaxa* and at the beginning of *uwwarapaxa*. *vyaFjana sandhi* must have a consonant character at the beginning of *uwwarapaxa* and there can be any character *svara* or *vyaFjana* at the end of the *pUrvapaxa*.

This classification helps us to apply the *sandhi* rules unambiguously to some extent. However, in some cases, we can see some overlapping of the rules of the *sandhi*. One such instance is $nAnu + illi = nAnilli$ $[I + here = I\ am\ here]$. Though the sandhi rule of *vakAra-Agama sandhi* explained by Sharma (2015) is applicable, we have to choose *lopa sandhi*, based on the usage. There are no research or reason present for the preference of *lopa sandhi* over the *Agama sandhi*. Due to this complexity, we have excluded the *Agama sandhi* in this paper.

The work on *sandhi* generation for Kannada has not been successfully carried out. There are *sandhi* generator tools developed for basic Malayalam by Kleenankandy (2014) and for Sanskrit by Amba (2002), Huet (1994) and many others. We may find some works in *sandhi* generation for other Indian languages (Nirmala and Kalpana, 2015) as well. The *sandhi* splitter tasks were carried out for Kannada by Shashirekha and Vanishree (2016), but generator was left out due to the complexity and ambiguity involved. We are addressing this issue by proposing a novel idea by defining the algorithm necessary to generate *sandhi* formation for the given input Kannada words using a rule based approach. The *sandhi* generator is useful for students who wish to learn the concept of euphonic change in Kannada and to researchers working on NLP applications for the tasks like morphological analysers and Machine Translation. The paper is organized as follows: Section 1 introduction, section 2 literature review, section 3 methodology, section 4 result analysis and section 5 conclusion and future scope followed by references.

2 Literature Survey

Kannada spell checker and *sandhi* splitter work was carried out by Murthy et al. (2017) making use of transliterated dictionaries which stored the words and affixes. The *Agama sandhi* splitter was developed by Shashirekha and Vanishree (2016) using a rule based approach with manual annotation of suitable words and their affixes. There are other Kannada *sandhi* splitter works but here we have reviewed a few as our focus is on generation than splitting. Kleenankandy (2014) has implemented *sandhi*-rule based compound word generator for Malayalam, the basic *sandhi* rules for Malayalam was addressed along with supplementary details for words to identify the *sandhi*. This work was semi-automatic and required human intervention to resolve ambiguities. Significant work on Sanskrit word segmentation was carried out by Huet (2003).

The disambiguation of a given word was performed using a rule-based approach.

Though there are *sandhi* generator tools developed for Sanskrit by Amba (2002), Huet (1994), Sachin Kumar and many others, we see that there has not been a successful *sandhi* generator for pure Kannada *sandhi*.

3 Methodology

The implementation of a basic *lopa* and *Adesha sandhi* using regular expression is presented in this section. We have excluded the internal *sandhi* (root + suffix) and considered only compound words and external *sandhi*. As we mentioned earlier in the introduction section, some rules of *Agama sandhi* overlaps with the rules of *lopa sandhi*. Let us consider the *yakAra-Agama sandhi* as an example - *maneV + alli = maneVyalli [house + locative case marker = in the house]*. The rules for *lopa sandhi* and *Agama sandhi* overlap here and there is no semantic information which can distinguish and avoid the overlap. By convention and practical usage, *maneVyalli* is the expected output, whereas due to the overlapping rules, our system generates *maneli* as the output. Though the word *maneli* is also correct in colloquial Kannada and the meaning is also exactly similar as *maneVyalli*, we may not find the word *maneli* in formal usage. Hence, we have skipped writing the algorithms to *Agama sandhi* at this stage.

The block diagram of the methodology is shown in Figure 1.

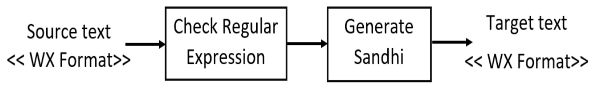


Figure 1: Block diagram of the system for *sandhi* generation

The user inputs two Kannada words in WX notation. The *Check Regular Expression* module will validate the input words to check if the words follow the pattern suitable to perform *lopa* or *Adesha sandhi*. If it follows the pattern, then the inputs are passed into *Generate Sandhi* module. In this module, the rules are defined. The rules to perform *lopa* and *Adesha sandhi* are different. We have referred mainly to the Kannada grammar book by Sharma (2015) and inspired by the Sanskrit Sandhi works of Amba (2002) and Huet (1994) to obtain these rules and implementation was performed following the same. The generated output is the *sandhi* word in WX notation. However, we may render the WX input in UTF-8 (*Kannada scripts*) to ease the process of learning or understanding the *sandhi* concept, using the existing standard transliteration schemes.

3.1 Algorithm for *lopa* and *Adesha Sandhi*

The algorithm for *lopa sandhi* and *Adesha sandhi* is shown in Algorithm 1. As we mentioned, we have followed rule based approach and is based on the morphological rules of Kannada by Sharma (2015). The characters are extracted using extract function as defined in Algorithm 2 and are joined with respect to the predefined morphological rules. The input words are analysed to check which *sandhi* can be applied. Once we decide on the *sandhi*, unification of characters can be done based on rules. No dictionary is used here and we assume that user has typed valid Kannada words as input. We would like to make use of a dictionary in the future to validate input words. As of now, if the rules are satisfied by the input words, the corresponding output is generated.

3.1.1 Definitions

Svara: Vowels in Kannada language - *a, A, i, I, u, U, q, eV, e, E, oV, o, O*
ktp: characters *k* or *w* or *p*

3.2 Implementation details

The text in WX form is checked against the regular expressions. There are four separate regular expressions two each for *lopa* and *Adesha sandhi*. For instance, in *lopa sandhi* the *pUrvapaxa* has to end with a *svara*. The corresponding regular expression is $[A-Za-z]*([aAiIuUqeEoO]|(eV)|(oV))\$$. We have made use of regular expression library[3] available for Python 3 to match these and implemented on Google Colab[4] platform.

4 Results and Analysis

We have checked in total for 386 unique pairs of Kannada words manually extracted from the data we requested from the work carried out by Reddy and Sharoff (2011). We were able to identify 255 pairs for *lopa sandhi* and 131 pairs for *Adesha sandhi*. The sample data is shown in table 1[5]. The

[3]https://docs.python.org/3/library/re.html
[4]https://colab.research.google.com
[5]We have not used diacritics in this table to highlight that the input to the system must be in WX notation.

Algorithm 1: Kannada *sandhi* generator for *lopa* and *Adesha sandhi*

```
Data: string1, string2
Result: sandhi,output_string
$len1 \leftarrow length(string1) - 1$
$len2 \leftarrow length(string2) - 1$
if string1 ends with svara then
    if string2 begins with svara then
        if string1 ends with 'eV' or 'oV' then
            $e1 \leftarrow extract(string1, 0, len1 - 2)$
        else
            $e1 \leftarrow extract(string1, 0, len1 - 1)$
        end
    $e2 \leftarrow string2$
    $k \leftarrow concatenate(e1, e2)$
    print "Lopa Sandhi", k
else if string1 contains only characters then
    if if string2 begins with ktp then
        if first_char(string2)='N' or 'n' then
            $temp \leftarrow extract(string1, 0, len1 - 1)$
            $e1 \leftarrow concatenate(temp, 'M')$
        else
            $e1 \leftarrow string1$
        end
    $e2 \leftarrow extract(string2, 1, len2)$
    $check \leftarrow string2[0]$
    if check='k' then
        $add \leftarrow 'g'$
    else if check='w' then
        $add \leftarrow 'x'$
    else
        $add \leftarrow 'b'$
    end
    $temp2 \leftarrow concatenate(e1, add)$
    $f \leftarrow concatenate(temp2, e2)$
    print "Adesha Sandhi", f
else
    print "Check your input"
end
```

generated words were examined by a Kannada linguistics expert for its accuracy.

The proposed work is useful in the generation of euphonic change without a dictionary and maybe applicable to other Dravidian languages. The summary of result and the accuracy is shown in table 2.

Algorithm 2: Extract substring

```
Data: string, start_len, end_len
Result: substring
$j \leftarrow 0$
for each i ←start_len to end_len do
    $m[j] \leftarrow concatenate(m, string[i])$
    $j \leftarrow j + 1$
end
$m[j] \leftarrow ' \backslash 0'$
```

Input1	Input2	Output string	Sandhi
nAnu [*I*]	illi [*here*]	nAnilli [*I am here*]	lopa
nanna [*me*]	iMxa [*by*]	nanniMxa [*by me*]	lopa
nanna [*my*]	Uru [*place*]	nannUru [*My place*]	lopa
kaliyaxe [*not*]	iruvuxu [*to learn*]	kaliyaxiruvuxu [*to not learn*]	lopa
xevaru [*God*]	iMxa [*by*]	xevariMxa [*by God*]	lopa
maneV [*house*]	keVlasa [*work*]	maneVgeVlasa [*house work*]	Adesha
adi [*foundation*]	kallu [*stone*]	adigallu [*foundation stone*]	Adesha
mE [*body*]	woVlYeV [*wash*]	mExoVlYeV [*wash the body*]	Adesha
hullu [*hay*]	kAvalu [*land*]	hullugAvalu [*hay land*]	Adesha
betta [*mountain*]	wAvareV [*lotus*]	bettaxAvareV [*mountain lotus*]	Adesha

Table 1: Sample output for *lopa* and *Adesha sandhi*

Sandhi	Test pairs	Correct output	Accuracy
Lopa	255	255	100%
Adesha	131	130	99.2%

Table 2: Summarized results for *lopa* and *Adesha sandhi*

In one of the instances of *Adesha sandhi*, "*kaN + kAvalu = kaNgAvalu [eye + security = surveillance*]", the generated output was *kaMgAvalu* which is not a valid *sandhied* word. In the case of *lopa sandhi*, all the generated outputs are valid outputs. Since the input words are not validated by a morphological analyser, any non-Kannada word which follows the pattern prescribed for *lopa* and *Adesha sandhi* will generate a *sandhi* output as per the rules is the major flaw in this tool. However, for a given input, no pair can undergo both *lopa* and *Adesha sandhi* at once. The evaluation of this task is performed manually by checking the generated output for the correct euphonic change and the type of that *sandhi*, and is verified by a senior linguist and a few native speakers as well.

5 Conclusion and Future Scope

A basic *sandhi* generator for *lopa* and *Adesha sandhi* is carried out in this work. This work can be helpful for beginners and teachers who are engaged in the Kannada language and trying to understand/teach the concept of *sandhi*. In future, we would like to enhance this work by adding the non-overlapping and unambiguous rules for *Agama sandhi*. We will include the dictionary in the process to check the validity of the input words. We shall make a user-friendly interface to interact with the system. The generated words can be used to enhance the vocabulary for performing language related tasks viz. Kannada Machine Translation(MT) systems, Kannada Computational Linguistics tools etc. in future.

Acknowledgments

The first author is thankful to Prof. S. A. Krishnaiah, National Trust for Computation and Archival of Oriental Media (NTC-AOM), Udupi for verifying the accuracy of the generated outputs, Mrs. Shwetha Rai, Department of Computer Science & Engineering, Manipal Institute of Technology, Manipal for inputs in designing the algorithms and Dr. Siva Reddy, Mila, McGill University for sharing the PoS data for Kannada. We thank the anonymous reviewers for their constructive suggestions.

References

Kulkarni Amba. 2002. Samsaadhanii. =https://sanskrit.uohyd.ac.in/scl. Accessed: 04-Dec-2021.

Rohit Gupta, Pulkit Goyal, and Sapan Diwakar. 2010. Transliteration among indian languages using wx notation. In *KONVENS*.

Gérard Huet. 1994. The sandhi engine. =https://sanskrit.inria.fr/DICO/sandhi.fr.html. Accessed: 04-Dec-2021.

Gérard Huet. 2003. Lexicon-directed segmentation and tagging of sanskrit. In *in XIIth World Sanskrit Conference*, pages 307–325.

Gérard Huet. 2009. Sanskrit segmentation. In *South Asian Languages Analysis, round table*.

Rev. F Kittel. 1920. *Shabdamanidarpanam, Kesiraja's Jewel Mirror of Grammar*. Kanarese Mission Book and Tract Depository.

Jeena Kleenankandy. 2014. Implementation of sandhi-rule based compound word generator for malayalam. *2014 Fourth International Conference on Advances in Computing and Communications*, pages 134–137.

Anil Kumar, V. Sheeba, and Amba P. Kulkarni. 2009. Sanskrit compound paraphrase generator.

S Rajashekara Murthy, A. N. Akshatha, Chandana G Upadhyaya, and P. Ramakanth Kumar. 2017. Kannada spell checker with sandhi splitter. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 950–956.

K. Nirmala and M. K. Kalpana. 2015. Modern thamizh sandhi rules generator in nlp. In *SOCO 2015*.

Siva Reddy and Serge Sharoff. 2011. Cross language pos taggers (and other tools) for indian languages: An experiment with kannada using telugu resources.

Dr. Girish Nath Jha Rajneesh Kumar Pandey Sachin Kumar, Diwakar Mani. Sanskrit sandhi generator. `http://sanskrit.jnu.ac.in/sandhi/gen.jsp`. Accessed: 04-Dec-2021.

Vidhwan N Ranganatha Sharma. 2015. *Hosagannada Vyakarana*. Kannada Sahitya Parishat.

H. L. Shashirekha and K. S. Vanishree. 2016. Rule based kannada agama sandhi splitter. In *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 549–553.

Data Augmentation for Low-Resource Named Entity Recognition Using Backtranslation

Usama Yaseen[1,2], Stefan Langer[1,2]
[1]Technology, Siemens AG Munich, Germany
[2]Center for Information and Language Processing, LMU Munich, Germany
{usama.yaseen,langer.stefan}@siemens.com

Abstract

The state of art natural language processing systems relies on sizable training datasets to achieve high performance. Lack of such datasets in the specialized low resource domains lead to suboptimal performance. In this work, we adapt backtranslation to generate high quality and linguistically diverse synthetic data for low-resource named entity recognition. We perform experiments on two datasets from the materials science (MaSciP) and biomedical (S800) domains. The empirical results demonstrate the effectiveness of our proposed augmentation strategy, particularly in the low-resource scenario.

1 Introduction

Most recently, various deep learning methods have demonstrated state of art performance for many natural language processing tasks such as text classification, sentiment analysis and named entity recognition. The availability of large training datasets is crucial to achieve this improved performance and avoid overfitting. However, in many real-world applications collecting such large training data is not possible. This is especially true for specialized domains, such as the material science or biomedical domain, where annotating data requires expert knowledge and is usually time-consuming and expensive.

Data augmentation (DA) (Simard et al., 1996) has been investigated to overcome this low resource problem. Label preserving synthetic data generation is widely used in computer vision (Krizhevsky et al., 2012; Ciresan et al., 2012; Fawzi et al., 2016) and speech domains (Schlüter and Grill, 2015; Ko et al., 2017). The discrete nature of language makes it difficult to adapt data augmentation strategies from computer vision and speech to natural language processing. Unlike computer vision, where hardcoded transformations (such as rotation, masking, cropping etc.) can be easily applied without changing the label semantics, the manipulation of a single word in a sentence could change its meaning.

Recently, there is an increased interest in applying data augmentation to natural language processing tasks. Most augmentation methods explore sentence-level tasks such as sentiment analysis (Liesting et al., 2021), text classification (Wei and Zou, 2019; Xie et al., 2019) and sentence-pair tasks such as natural language inference (Min et al., 2020) and machine translation (Wang et al., 2018). The augmentation methods either employ heuristics such as word replacement (Zhang et al., 2015; Wang et al., 2018; Cai et al., 2020), word swap (Sahin and Steedman, 2018; Min et al., 2020) or random deletion (Wei and Zou, 2019) to generate augmented instances by manipulating a few words in the original sentence; or generate completely artificial instances via sampling from generative models such as variational autoencoders (Yoo et al., 2019; Mesbah et al., 2019) or backtranslation models (Yu et al., 2018; Iyyer et al., 2018).

The sequence labelling tasks such as named entity recognition (NER) and part-of-speech tagging (POS) involves prediction at the token level. This makes applying token-level transformation difficult as such manipulations may change the corresponding token level label. The existing DA methods for sequence labelling uses dependency tree morphing (Sahin and Steedman, 2018), MIXUP (Zhang et al., 2018) to generate queried samples in the active learning scenario (Zhang et al., 2020), sample novel sequences from a trained language model (Ding et al., 2020) and apply pre-defined heuristics such as label-wise token and synonym replacement (Dai and Adel, 2020). The existing sequence labelling DA methods are limiting as they: a). rely on linguistics resources like dependency parser or WordNet b). involves training a language model c).

generate grammatically incoherent sequences d). cannot generate linguistically diverse sequences.

Motivated by the advancements in machine translation and the availability of high-quality machine translation systems (He, 2015; Wu et al., 2016; Junczys-Dowmunt, 2019), in this work we adapt backtranslation to the task of NER. Backtranslation (BT) can automatically generate diverse paraphrases of a sentence or a phrase by naturally injecting linguistic variations. The injected linguistic variations can be further diversified by introducing layers of intermediate language translations. In this work, we generate paraphrases of one or several phrases in a sentence. We empirically demonstrate the effectiveness of our proposed method on two domain-specific NER datasets.

2 Related Work

There is an abundance of recent work on DA methods for NLP tasks, we refer the readers to Feng et al. for an extensive survey. In this section we narrow our focus to existing DA methods for sequence labelling tasks like NER and POS. We categorize existing DA methods for sequence labelling into two categories:

Rule-based: DA primitives, which use predefined easy-to-compute transformations. We briefly describe six of such transformations proposed in the existing work:

(a) *NER::Label-wise token replacement (LwTR):* Replace a token with another token of the same entity type at random (Dai and Adel, 2020).

(b) *NER::Synonym replacement (SR):* Replace a token with one of its synonyms retrieved from WordNet at random (Dai and Adel, 2020).

(c) *NER::Mention replacement (MR):* Replace an entity mention with another entity mention of the same entity type at random (Dai and Adel, 2020).

(d) *NER::Shuffle within segments (SiS):* Divide the sequence of tokens into segments of the same label and then randomly shuffle the order of segments (Dai and Adel, 2020).

(e) *POS::Crop Sentences:* Given a dependency tree of the sentence, "crop" a sentence by removing dependency links (Sahin and Steedman, 2018).

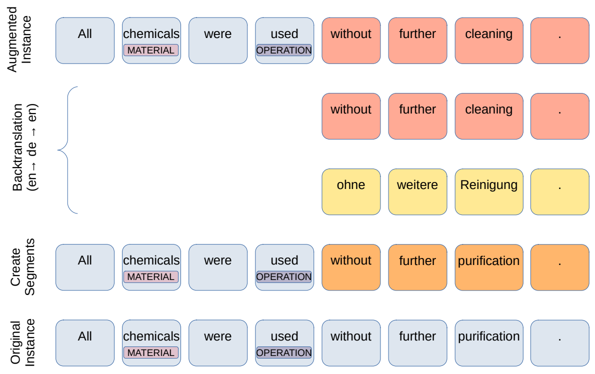


Figure 1: An illustration of data augmentation via *backtranslation* for NER. Note that backtranslation is only applied to the context around the entity mentions. Here the entity mention context is first translated to German and then back to English using an off-the-shelf machine translation system. The backtranslation results in a paraphrase of the original entity mention context. The original entity mention context is replaced with backtranslated context to create the augmented data instance.

(f) *POS::Rotate Sentences:* Given a dependency tree of the sentence, "rotate" a sentence by moving the tree fragments around the root (Sahin and Steedman, 2018).

Generative models: The existing work uses pre-trained language models to generate either part of the sequence or the entire sequence with the corresponding NER tags. Kang et al. proposed *Filtered BERT* which randomly masks one or several tokens in the original sentence and let BERT (Devlin et al., 2019) predict the masked token. The augmentation is only accepted if the cosine similarity of the word embeddings (computed using fastText embeddings (Bojanowski et al., 2017)) of the original token and the predicted masked token is above a certain threshold. Ding et al. propose a two-step DA process DAGA. First, a shallow language model is trained over linearized sequences of tags and words. Second, sequences are sampled from this language model and delinearized to create new examples.

3 Data Augmentation via Backtranslation

Figure 1 illustrates an example of data augmentation using *backtranslation* for NER with German as a pivot language. In a nutshell, the algorithm consists of three steps. First, the input token sequence is split into segments of the same label; thus, each segment corresponds to either the entity mention or the context around the entity mention. Note that only context around the entity mention is a candidate for the backtranslation. Second, the validity of the segment is determined based on the length of the segment, we only consider segments with three or more tokens as a valid segment for backtranslation. As a final step, the segment tokens are translated to the intermediate pivot language(s) and finally back to the source language; the original segment tokens are replaced with the backtranslated tokens and thus we obtain the augmentation of the original input token sequence. In practice, we use a binomial distribution to randomly decide whether the segment should be backtranslated. Since only the context around the entity mention is backtranslated, it is straightforward to adjust the corresponding BIO-label sequence accordingly for the backtranslated text.

4 Experiments and Results

4.1 Datasets

We empirically evaluate backtranslation for NER on two English datasets from the materials science and biomedical domains: MaSciP (Mysore et al., 2019)[1] and S800 (Pafilis et al., 2013)[2]. MaSciP contains synthesis procedures annotated with synthesis operations and their typed arguments. S800 consists of PubMed abstracts annotated for organism mentions. We use the original train-dev-test split provided by the authors.

We simulate low-resource setting as proposed by Dai and Adel; we select 50, 150, 500 sentences from the training set to create the corresponding small, medium and large training sets (denoted as S, M, L in Table 1, whereas the complete training set is denoted as F). Data augmentation is only applied on the training set without altering the development and test set.

4.2 NER Model

We follow the standard approach of modelling the NER task as a sequence labelling task. The mainstream sequence labelling models for NER employ the neural-based encoder and an output tagging component. The typical choice of the encoder is a sequence model such as LSTM (Hochreiter and Schmidhuber, 1997) or more recently a sequence encoder such as Transformer (Vaswani et al., 2017);

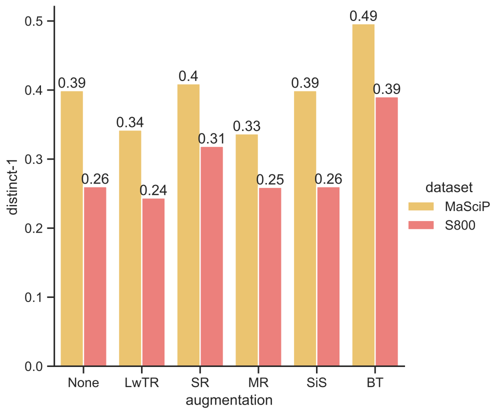


Figure 2: The diversity statistics of various augmentation techniques across the datasets.

the output tagging component is usually a conditional random field layer (Lafferty et al., 2001) to model dependencies between neighbouring labels.

We employed the standard *BiLSTM-CRF* model (Lample et al., 2016) as our backbone model. We experimented with context-independent *GloVe* embeddings (Pennington et al., 2014) as well as state-of-the-art contextualized *BERT* embeddings (Devlin et al., 2019). We employed *SciBERT* (Beltagy et al., 2019), which is based on the BERT model pretrained on scientific publications; our preliminary experiments suggest that SciBERT achieves better performance than BERT. The superiority of domain-specific BERT models on downstream tasks has been observed by existing studies (Gururangan et al., 2020; Dai and Adel, 2020).

We report the micro-average F_1 score as an evaluation metric. We employ early stopping and report the F_1 score on the test set using the best performant model on the development set.

4.3 Backtranslation Models

We employed the Huggingface's Transformers library (Wolf et al., 2020) port of the pretrained English↔German models (Ng et al., 2019)[3,4] as the underlying backtranslation models for all our experiments.

4.4 Hyperparameters

Following existing work (Dai and Adel, 2020), we tune the number of augmentation instances per training instance from a list of numbers:

[1]https://github.com/olivettigroup/annotated-materials-syntheses
[2]https://github.com/spyysalo/s800

[3]https://huggingface.co/facebook/wmt19-en-de
[4]https://huggingface.co/facebook/wmt19-de-en

Embeddings	Method	MaSciP				Δ	S800				Δ
		S	M	L	F		S	M	L	F	
Glove	None	48.52± 3.5	67.98± 0.5	73.02± 0.8	75.37± 0.3		12.24± 1.6	21.61± 0.7	49.99± 2.6	60.44± 1.4	
	LwTR	61.95± 1.3	68.04± 0.7	75.05± 0.3	75.32± 0.2	2.9	17.37± 0.4	41.19± 1.3	50.93± 1.8	62.46± 1.2	6.9
	SR	**63.91± 1.6**	69.44± 0.7	75.10± 0.4	76.95± 0.8	4.6	17.83± 1.3	43.86± 1.1	57.76± 0.2	65.28± 0.5	10.1
	MR	63.46± 0.3	69.64± 0.7	75.08± 0.4	76.33± 1.0	4.6	17.86± 2.4	43.90± 0.8	56.70± 0.9	65.34± 0.6	9.9
	SiS	63.63± 1.1	69.60± 0.3	73.35± 0.2	**77.36± 0.3**	4.6	17.17± 1.7	44.36± 0.2	56.80± 0.9	64.93± 0.2	9.7
	BT	63.66± 0.6	**69.67± 0.1**	**75.22± 0.2**	76.85± 0.4	**4.6**	**31.06± 1.7**	**47.82± 1.2**	**58.86± 1.0**	**66.89± 0.3**	**15.1**
SciBERT	None	61.89± 1.3	71.76± 0.6	78.52± 0.1	79.91± 0.1		39.78± 1.6	51.15± 1.6	64.08± 0.8	72.73± 0.9	
	LwTR	66.88± 1.4	73.40± 1.1	77.83± 0.1	77.51± 3.0	0.9	41.37± 0.4	51.76± 1.0	64.97± 1.6	71.34± 0.1	0.4
	SR	67.07± 0.8	74.56± 0.3	78.47± 0.4	79.71± 0.3	1.9	40.24± 1.2	**53.68± 0.4**	62.98± 1.4	71.77± 0.6	0.2
	MR	67.65± 1.0	74.60± 1.3	78.04± 1.1	79.57± 0.6	1.9	41.89± 1.4	53.24± 1.3	66.56± 1.2	70.87± 0.5	1.2
	SiS	66.87± 2.9	73.40± 1.5	**78.95± 0.6**	79.79± 0.5	1.7	41.57± 1.8	51.83± 0.7	65.16± 1.0	71.20± 0.6	0.5
	BT	**70.11± 0.8**	**75.86± 0.8**	78.92± 0.2	**80.30± 0.5**	**3.3**	**44.60± 1.0**	53.22± 1.3	**66.76± 1.1**	**72.92± 0.2**	**2.4**

Table 1: F1-score on test sets using different subsets of the training set. Here: **S**, **M**, **L** and **F** refer to *small* (50 instances), *medium* (150 instances), *large* (500 instances) and *full* (all instances) set. We repeat all experiments three times with different seeds. Mean values and standard deviations are reported. Δ column shows the averaged improvement due to data augmentation for each embedding type across the datasets.

$\{1, 3, 6, 10\}$. When the complete dataset is used, this tuning list is reduced to: $\{1, 2, 3\}$. We also tune the probability value p of the beta distribution which is used to decide if the segment in a sequence should be backtranslated. It is searched over a list of numbers: $\{0.1, 0.3, 0.5, 0.7\}$. We perform a grid search over these two hyperparameters to find their best combination on the development set.

4.5 Results

We report the performance of various augmentation techniques on the test sets in Table 1. For the most part, all data augmentation techniques improve over the baseline; backtranslation results in the biggest average improvement for both context-independent *GloVe* and contextualized *SciBERT* embeddings under different data usage percentiles. We attribute the improved performance of backtranslation to the generation of linguistically diverse and meaning-preserving *entity mention contexts* to enable better generalization of the underlying NER model.

The data augmentation techniques contribute to the biggest improvement in performance when the training sets are small, this effect is reduced as the training sets get larger (see columns **S** vs **F** in Table 1). The augmentation on the complete training set even decreases the performances for some augmentation techniques. The performance impact of data augmentation on varying sizes of training sets has also been observed in the existing work (Fadaee et al., 2017; Dai and Adel, 2020; Ding et al., 2020).

We also investigate the effectiveness of data augmentation techniques on the mainstream contextualized (pretrained *SciBERT*) embeddings. All the augmentation techniques especially backtranslation result in better performance when compared to the baseline. However, the average performance improvement due to data augmentation with SciBERT embeddings is lower as compared to the GloVe embeddings.

To quantitatively measure the diversity introduced by various augmentation techniques, we report *distinct-1* (Li et al., 2016) in Figure 2. Distinct-1 quantifies the intra-text diversity based on distinct unigrams in each sentence, the value is scaled by the total number of tokens in the sentence to avoid favouring long sentences. Backtranslation yield the highest level of unigram diversity, this is not very surprising as backtranslation is known to generate diverse linguistic variations.

5 Conclusion

In this paper, we adapt backtranslation to the token-level sequence tagging task of NER. We show that backtranslation can generate high-quality coherent and linguistically diverse synthetic data for NER. The experiments on two domain-specific datasets demonstrate the effectiveness of backtranslation as a competitive data augmentation strategy for NER.

Acknowledgments

This research was supported by the Federal Ministry for Economic Affairs and Energy (Bundesministerium für Wirtschaft und Energie: `https://bmwi.de`), grant 01MD19003E (PLASS: `https://plass.io`) at Siemens AG (Technology- Machine Intelligence), Munich Germany.

References

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 3613–3618. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomás Mikolov. 2017. Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguistics*, 5:135–146.

Hengyi Cai, Hongshen Chen, Yonghao Song, Cheng Zhang, Xiaofang Zhao, and Dawei Yin. 2020. Data manipulation: Towards effective instance learning for neural dialogue generation via learning to augment and reweight. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 6334–6343. Association for Computational Linguistics.

Dan C. Ciresan, Ueli Meier, and Jürgen Schmidhuber. 2012. Multi-column deep neural networks for image classification. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pages 3642–3649. IEEE Computer Society.

Xiang Dai and Heike Adel. 2020. An analysis of simple data augmentation for named entity recognition. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 3861–3867. International Committee on 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Bosheng Ding, Linlin Liu, Lidong Bing, Canasai Kruengkrai, Thien Hai Nguyen, Shafiq Joty, Luo Si, and Chunyan Miao. 2020. DAGA: Data augmentation with a generation approach for low-resource tagging tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6045–6057, Online. Association for Computational Linguistics.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. 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 567–573. Association for Computational Linguistics.

Alhussein Fawzi, Horst Samulowitz, Deepak S. Turaga, and Pascal Frossard. 2016. Adaptive data augmentation for image classification. In *2016 IEEE International Conference on Image Processing, ICIP 2016, Phoenix, AZ, USA, September 25-28, 2016*, pages 3688–3692. IEEE.

Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard H. Hovy. 2021. A survey of data augmentation approaches for NLP. *CoRR*, abs/2105.03075.

Suchin Gururangan, Ana Marasovic, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8342–8360. Association for Computational Linguistics.

Zhongjun He. 2015. Baidu translate: Research and products. In *Proceedings of the Fourth Workshop on Hybrid Approaches to Translation, HyTra@ACL 2015, July 31, 2015, Beijing, China*, pages 61–62. The Association for Computer Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. 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 1875–1885. Association for Computational Linguistics.

Marcin Junczys-Dowmunt. 2019. Microsoft translator at WMT 2019: Towards large-scale document-level neural machine translation. In *Proceedings of the Fourth Conference on Machine Translation, WMT 2019, Florence, Italy, August 1-2, 2019 - Volume 2: Shared Task Papers, Day 1*, pages 225–233. Association for Computational Linguistics.

Min Kang, Kye Lee, and Youngho Lee. 2021. Filtered bert: Similarity filter-based augmentation with bidirectional transfer learning for protected health information prediction in clinical documents. *Applied Sciences*, 11:3668.

Tom Ko, Vijayaditya Peddinti, Daniel Povey, Michael L. Seltzer, and Sanjeev Khudanpur. 2017. A study on data augmentation of reverberant speech for robust speech recognition. In *2017 IEEE International Conference on Acoustics, Speech and*

Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017, pages 5220–5224. IEEE.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114.

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 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 282–289. Morgan Kaufmann.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 260–270. The Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 110–119. The Association for Computational Linguistics.

Tomas Liesting, Flavius Frasincar, and Maria Mihaela Trusca. 2021. Data augmentation in a hybrid approach for aspect-based sentiment analysis. In *SAC '21: The 36th ACM/SIGAPP Symposium on Applied Computing, Virtual Event, Republic of Korea, March 22-26, 2021*, pages 828–835. ACM.

Sepideh Mesbah, Jie Yang, Robert-Jan Sips, Manuel Valle Torre, Christoph Lofi, Alessandro Bozzon, and Geert-Jan Houben. 2019. Training data augmentation for detecting adverse drug reactions in user-generated content. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 2349–2359. Association for Computational Linguistics.

Junghyun Min, R. Thomas McCoy, Dipanjan Das, Emily Pitler, and Tal Linzen. 2020. Syntactic data augmentation increases robustness to inference heuristics. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2339–2352. Association for Computational Linguistics.

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, LAW@ACL 2019, Florence, Italy, August 1, 2019*, pages 56–64. Association for Computational Linguistics.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook fair's WMT19 news translation task submission. In *Proceedings of the Fourth Conference on Machine Translation, WMT 2019, Florence, Italy, August 1-2, 2019 - Volume 2: Shared Task Papers, Day 1*, pages 314–319. Association for Computational Linguistics.

Evangelos Pafilis, Sune P. Frankild, Lucia Fanini, Sarah Faulwetter, Christina Pavloudi, Aikaterini Vasileiadou, Christos Arvanitidis, and Lars Juhl Jensen. 2013. The species and organisms resources for fast and accurate identification of taxonomic names in text. *PLoS ONE*, 8(6):1–6.

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 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL.

Gözde Gül Sahin and Mark Steedman. 2018. Data augmentation via dependency tree morphing for low-resource languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 5004–5009. Association for Computational Linguistics.

Jan Schlüter and Thomas Grill. 2015. Exploring data augmentation for improved singing voice detection with neural networks. In *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*, pages 121–126.

Patrice Y. Simard, Yann LeCun, John S. Denker, and Bernard Victorri. 1996. Transformation invariance in pattern recognition-tangent distance and tangent propagation. In Genevieve B. Orr and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*, pages 239–27. 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. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural*

Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 5998–6008.

Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. 2018. Switchout: an efficient data augmentation algorithm for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 856–861. Association for Computational Linguistics.

Jason W. Wei and Kai Zou. 2019. EDA: easy data augmentation techniques for boosting performance on text classification 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 2019, Hong Kong, China, November 3-7, 2019*, pages 6381–6387. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 38–45. Association for Computational Linguistics.

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, Lukasz 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. *CoRR*, abs/1609.08144.

Qizhe Xie, Zihang Dai, Eduard H. Hovy, Minh-Thang Luong, and Quoc V. Le. 2019. Unsupervised data augmentation. *CoRR*, abs/1904.12848.

Kang Min Yoo, Youhyun Shin, and Sang-goo Lee. 2019. Data augmentation for spoken language understanding via joint variational generation. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 7402–7409. AAAI Press.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Rongzhi Zhang, Yue Yu, and Chao Zhang. 2020. Seqmix: Augmenting active sequence labeling via sequence mixup. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 8566–8579. Association for Computational Linguistics.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657.

Semantics of Spatio-Directional Geometric Terms of Indian Languages

Sukhada[1], **Soma Paul**[2], **Rahul Kumar**[3], **Karthik Puranik**[4]
[1]IIT (BHU), Varanasi, [2]IIIT-Hyderabad,[3]University of Delhi [4]IIIT Tiruchirappalli
sukhada.hss@iitbhu.ac.in, soma@iiit.ac.in,
sadarpurrahul1@gmail.com, karthikp18c@iiitt.ac.in

Abstract

This paper examines widely prevalent yet little-studied expressions in Indian languages which are known as geometrical terms because "they engage locations along the axes of the reference object". These terms are *andara* (inside), *bāhara* (outside), *āge* (in front of), *sāmane* (in front of), *pīche* (back), *ūpara* (above/over), *nīce* (under/below), *dāyeṃ* (right), *bāyeṃ* (left), *pāsa* (near), *dūra* (away/far) in Hindi. The way these terms have been interpreted by the scholars of Hindi language and handled in the Hindi Dependency treebank is misleading. This paper proposes **an alternative analysis of these terms focusing on their triple – nominal, modifier and relational - functions and presents abstract semantic representations** of these terms following the proposed analysis. The semantic representation will be explicit, unambiguous abstract and therefore universal in nature. The correspondence of these terms in Bangla and Kannada are also identified. Disambiguation of geometric terms will facilitate parsing and machine translation especially from Indian Language to English because these geometric terms of Indian languages are variedly translated in English depending on context.

1 Introduction

Geometric terms that "engage locations along the axes of the reference object" (Landau, 2017) play multiple roles - relational, nominal, modifier - in Indian languages and their equivalents in English can vary based on their functional role. For example, the geometric term *nīce* in (1-a) and (1-b) is translated differently into English depending on the function of the word in the given two contexts:

(1) a. meja ke nīce
table.SG GEN under
cyūiṃgama
Chewing gum.SG.3.NOM
cipakī hai
stick.PR.SG.3
'There is a chewing gum stuck **under** the table'.

b. nīce jāo
down go.IMP.SG
'Go **down**'.

In (1-a), the term *nīce* indicates a spatio-directional[1] relation between *meja* (table) and *cyūiṃgama* (chewing gum) while in (1-b), nīce denotes downward location. Thus, in (1-a), *nīce* (under) is a relational marker that specifies the geometric position of 'chewing gum' with respect to the 'table' while in (1-b) *nīce* (down) indicates location, thus fulfilling a nominal function. Interestingly there are contexts where these terms inadvertently appear to be relational markers which they are not. For such cases we attribute a third role to them, the role of a modifier. For example, in (2) the term nīce is a spatial modification of the location *peṭī* (box). The location of 'books' is the box and the position of the box is specified by the geometric term *nīce*. The reference object *meja* (table) with respect to which *nīce* has to be interpreted is present in (2-a). In (2-b), we show that *nīce* can even take the adjectival suffix, thus making its modifier role clearer.

(2) a. meja ke nīce peṭī meṃ
table.SG GEN under box.SG LOC
pustaka rakhī hai
book.SG.NOM keep.PR.SG.3
'The book is kept in the box under the table.'

b. meja ke nīcevālī peṭī meṃ
table.SG GEN under.ADJ box.SG LOC
pustaka rakhī hai
book.SG.NOM keep.PR.SG.3
'The book is kept in the box that is under the table.'

[1]Spatio-directional refers to spatial cum directional semantics

Thus, we identify three roles of the geometric terms in Indian languages:

1. A spatial noun denoting a place as in (1-b)
2. A geometric spatio-directional relation marker as in (1-a).
3. A geometric spatio-directional modifier of a noun as in (2-a)

In this paper, we propose semantic interpretation of geometric terms and present an abstract semantic representation for them. The semantic representation will be explicit, unambiguous, abstract and therefore universal in nature. Such enquiry is significant not just for Natural Language Understanding, but also in the context of language transfer. We have shown in (1) that English, for example, uses a preposition when the geometric term is a relational marker, otherwise the language uses adverbs as in (1-b) while Hindi and other Indian languages such as Bangla and Kannada use the same lexical item in both (1-a) and (1-b).

The paper is divided into the following sections. Section 2 introduces the equivalents of Hindi spatio-directional geometric terms in Bangla and Kannada. Section 3 presents the semantic interpretation of these terms. Section 4 presents semantic representation of these terms and shows how such representation captures information explicitly and unambiguously, which are the characteristic features of any efficient semantic representation system. Finally, we will present the design of a Geometric terms Search Interface with annotation facility integrated in section 5.

2 Geometric Terms in Hindi, Bangla and Kannada

Geometric terms under considerations are *andara* (inside), *bāhara* (outside), *āge* (in front of), *sāmane* (in front of), *pīche* (back), *ūpara* (above/over), *nīce* (under/below), *dāyeṃ* (right), *bāyeṃ* (left), *pāsa* (near), *dūra* (away/far) in Hindi. There are some more geometric terms which are used in Hindi such as *āra-pāra* (across), *ora* (towards) which are purely relational markers and hence we keep them out of scope of this paper.

Table-1 presents equivalent lexical terms in Hindi, Bangla, Kannada and English for the relational variant of these terms and their morpho-lexical properties. Most of the English terms apart from 'right', 'left', 'inside' and 'outside' are spatial prepositions which are indeclinable.

Table-2 gives a quick comparative study of these lexical items in Hindi, Bangla and Kannada. The gloss for each example given in Table-2 is given in Appendix-1.

Fortis and Fagard following Levinson et al. have shown that relational nouns (spatial nominals) in Japanese and Korean follow a structure [Ground-GEN Spatial_Nominal-PostP] which, we see, is quite similar to Indian languages.

3 Semantic Interpretation of Geometric Terms

Talmy (1983) has introduced Figure-Ground geometric sense[6]"to refer to the located vs locating entity". Figure is "the object which is considered as moving or located with respect to another object" and Ground is "the object with respect to which a first is considered as moving or located" (Talmy, 1983) in the context of spatial configuration. Ground is alternatively referred to as a Reference object.

(3)

laḍakā$_{fig}$	ghara$_{gr}$	ke	bāhara
boy.SG.NOM	house	GEN	outside
khaḍā hai			
stand.PR.SG			

'The boy is standing outside the house.'

In Indian Grammatical Tradition, the relation is described between *ādhāra* (Ground) and *ādheya* (Figure). Two kinds of relations between *ādhāra* and *ādheya* have been identified in the literature which is very much relevant for our interpretation of geometric terms. This analysis also explains data recorded by Talmy in a different light that fits to our multirole analysis of these terms. The two relations are *saṃyoga-sambandha* (contact by touch) and *sāmīpya-sambandha* (contact by proximity) Shastri (1926); Subramania Iyer (1971). When there is a temporary physical contact *(saṃyoga)* between the *ādhāra* and the *ādheya* then the spatial relationship is said to have a *saṃyoga-sambandha*.

(4)

pakṣī	peḍa	ke	ūpara	baiṭhā hai	/
bird.SG.NOM	tree	GEN	above	sit.PR.SG.3	/
pakṣī	peḍa para	baiṭhā hai			
bird.SG.NOM	tree.LOC	sit.PR.SG.3			

'The bird is sitting on the tree.'

[2]The concept of Figure and Ground are borrowed from gestalt psychology to linguistics.

Hi.	Hi. form	Ba.	Ba. form	Ka.	Ka. form	En.	Example	En. Translation
andara	Indcl[2]	bhitare	bhitare[3]+ e[4]	oḷage	oḷa +akke	inside	ḍibbā thaile ke andara hai.	The box is inside the bag.
bāhara	Indcl	bāhira, bāire	bāira[5]+e	horage	hora +akke	outside	ghara ke bāhara nāma paṭṭī lagī hai.	The nameplate is outside the house.
āge	Indcl	āge	āga + e	mumde	Indcl	In front of	gāḍī ke āge nambara pleṭa nahīṃ hai.	There is no number plate in front of the car.
sāmane	Indcl	sāmane	Indcl	edhuru	Indcl	In front of	ye pirāmiḍa itihāsa kī eka alaga hī duniyā ko hamāre sāmane rakhate haiṃ.	These pyramids put a different kind of world before us.
pīche	Indcl	pichane	pichana +e	himdhe	Indcl	behind	gāḍī ke pīche nambara pleṭa nahīṃ hai.	There is no number plate behind the car.
ūpara	Indcl	opara, opore	opara +e	mele	Indcl	Above/ over/ on	meja ke ūpara kapa hai	The cup is on the table.
nīce	Indcl	nīce	nīca +e	keḷage	keḷa +akke	under	meja ke nīce cyūiṃgama cipakī hai.	The chewing gum is stuck under the table.
dāyeṃ	Indcl	dāine	Indcl	balakke	bala +akke	right	paidala yātrī saḍaka ke dāyīṃ ora caleṃ	Pedestrians should walk on the right side of the road.
bāyeṃ	Indcl	bāṃye	bāṃ +e	edakke	eda +akke	left	gāḍiyāṃ saḍaka ke bāyīṃ ora caleṃ	Cars should go on the left side of the road.
cāroṃ ora	Indcl	cāradike, cāridike	cāradika +e, cāridika +e	suttalu	sutta +alu	around	ribana momabattī ke cāroṃ ora baṃdhā huā hai.	The ribbon is tied around the candle.
bīca	Indcl	mājhakhāne	mājhakhāna +e	madhya	Indcl	between	nāva samudra ke bīca taira rahī hai.	The boat is floating in the middle of the ocean.
pāsa	Indcl	pāśe, kāche	pāśa +e, kācha +e	hattira	Indcl	near	hara mausama meṃ govā ke pāsa āpako parosane ke lie kucha khāsa hai.	In every weather Goa has something special to serve you.
dūra	Indcl	dūre	dūra +e	dūra	Indcl	far	-	-

Table 1: Morpho-lexical properties of relational variants of Geometric Terms in Hindi, Bangla, Kannada and English

[3]Indeclinable : indcl

[4]All the roots to which -e suffix has been added has an independent nominal occurrence in present Bangla

[5]-e is the locative marker in Bangla

[6]bāira ¡ bāhira (sadhu form of Bangla) does not have lexical

Morpho-syntax of geometric terms	Hindi	Bangla	Kannada
Nominal use	Frozen expressions: *ladakā pīche se āyā*	Bare forms with exceptions dāine and sāmane which are frozen: *chele-ṭā pechon theke elo*	Bare forms to which suffixes can be added: *huḍuga him̐dininm̐da bam̐danu*
Relational marker use	Frozen expressions. 'ke' variant of the genitive marker preceeds the relational marker: *meja ke nīche cyūiṃgama cipakī hai*	The '-e' suffix is added to the root form. Genitive suffix is added to the reference object: *ṭebil-er nīc-e cuing-gum āṭake āche*	The '-akke' suffix is added to the root form. Genitive suffix is added to the reference object: *ṭebal-ina keḷage cyūiṃgama am̐ṭide*
Modifier	The genitive marker optionally follows the geometric form: *ghara ke pīche (ke) bāgīce meṃ baḍe baḍe peḍa haiṃ*	*bāriṛ pechone(r) bāgāne baro baro gāch āche*	The genitive marker precedes and follows the geometric form: *maneya hindina tōṭadalli doḍḍa maragaḷive*

Table 2: A morpho-syntactic properties of geometric items in Hindi, Bangla and Kannada

When the *ādheya* does not touch the *ādhāra* but stays in proximity with *ādhāra* then that spatial relation is said to have *sāmīpya-sambandha*.

(5) peḍa ke ūpara cāṃda
tree.SG GEN above moon.SG.NOM
ā cukā hai
appear.PR.SG.3
'The moon has appeared above the tree.'

In order to localize a Figure with reference to Ground, Talmy has identified three kinds of expressions:

1. The expression indicates the Figure in touch with the Ground – same as *saṃyoga-sambandha*. Talmy observes that "the part of the Ground thus named is treated a regular noun" and usually occurs after 'the' in such cases,

   (6) The mosaic is on the back of the church.

   (7) The boy is in the front of the line.

2. The expression indicates the Ground's part to indicate 'immediate adjacency' – similar to *sāmīpya-sambandha*. Talmy has observed that in such cases in English, there is no 'the' before the geometric terms:

   (8) The bike is in back of/behind the church.

   (9) The police officer is in front of the line.

We omit the third type of expression here because they are poorly represented in English as Talmy has pointed out and do not come under the scope of this paper.

Given the above understanding, we propose two possible situations:

1. The Figure is located in a locus that is in 'part-whole' relation with the reference object,

   (10) peḍa ke ūpara pakṣī hai
   tree GEN on bird.NOM be.PR.SG
   'The bird is on the tree.'

   (11) The bike is behind the church (sentence (8) is repeated here)

2. The Figure is located in a locus that indicates a space denoted by the geometric term with respect to the reference object.

   (12) peḍa ke ūpara cāṃda
   tree GEN above moon.SG.NOM
   hai
   be.PR.SG
   'The moon is above the tree.'

   (13) The mosaic is on the back of the church (sentence (6) is repeated here)

attestation in the language

In case of 1, the geometric term mainly denotes the spatio-directional relation between the Figure and the reference object. The directional configuration of the Figure ‘bike’, for example in (8) , with respect to the reference object ‘church’ can be front/back/behind/ near and so on. Hence, this is the relational marker interpretation of the geometric terms.

In case of 2, on the other hand, the geometric term specifies a location. The location, although underspecified, indicates either directional or spatial position of the Figure. For example, the position of ‘*cāṃda*’ (moon) in (12) is in a space which is above the ‘peḍa’ (tree). This is the nominal use of the geometric terms. Such terms can be modified just like any other noun as shown in (14)-(17).

(14)

peḍa	**ke**	**ṭhīka**	**ūpara**	cāṃda	hai
tree.SG	GEN	right	above	moon.NOM	be.PR.SG

‘The moon is right above the tree.’

(15)

ghara	**se**	**4 kilomīṭara**	**dūra**	eka	mandira	hai
house.SG	ABL	4 kilometre	away	a	temple.NOM	be.PR.SG

‘There is a temple 4 kilometres away from the house.’

(16)

mandaura	**se**	**eka**	**mīla**	**nīce**	eka	choṭā	saṃgha	hai,
mandaura.NNP	ABL	a	mile	below	a	small	sangha.NOM	be.PR.SG

‘There is a small sangha a mile below the mandaura.’

(17)

kāṃkera	**se**	**22 kilomīṭara**	**āge**	śurū hotī hai	keśakāla	kī	manorama	ghāṭī
Kanker.NNP	ABL	22 kilometre	ahead	start.PR.SG	Keshkal.NNP	GEN	charming	valley.NOM

‘The charming valley of Keshkal starts 22 kilometres ahead of Kanker.’

4 Abstract Semantic Representation of expressions with Geometric terms

Keeping in tune with our analysis given in section 3, we will present here an abstract semantic representation of the geometric terms. Before we present our representation, we will examine how these terms have been represented in the Hindi dependency treebank. In Hindi grammar, these geometric terms have been considered as part of complex post-positions (Bharati et al., 1995; Kachru, 2006; Koul, 2008; Bharati et al., 2009). In Hindi dependency treebank, however, they are lexically marked as Noun-Space-Time (NST) thus distinguishing them from other post-positions which are tagged as PSP[7] (Bharati et al., 2007). The tag NST indicates that these lexical items are nouns denoting space and time, but their role in the context is not that of a noun. The semantic relation that has been annotated for the sentence “*laḍakā ghara ke bāhara khaḍā hai.*” where ‘*bāhara*’ is an NST as shown in Figure-1.

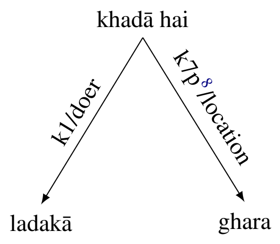


Figure 1: Basic dependency tree for sentence: *laḍakā ghara ke bāhara khaḍā hai.*

In dependency tree structures the content words are represented as nodes, the verb is the head of the tree (in case of simple sentences) and relational markers are all semantic labels on the edges connecting the nodes as shown in Figure-1. The convention of Hindi Dependency treebank is that the syntactico-semantic relations are marked among the chunks. Each chunk has a head and the relation is to be understood between the heads of the chunk. The meaning of this structure is that ‘*ke_bāhara*’ (which is annotated as PSP_NST at the POS level) is a relational marker and conveys ‘*deśādhikaraṇa*’ (locative) information. This analysis implies ‘*ghara*’ (house) the location where the boy stands. This is a misleading analysis because the boy does not stand at the house rather he stands in a place which is outside the house. ‘*bāhara*’ conveys that information but the analysis in the treebank does not explicitly capture that.

According to our interpretation of geometric terms, the representation of relational and nominal/modificational variants are postulated differently. Figure-2 presents the relational marker variant of geometric terms. For the sentence ‘*pakṣī peḍa ke ūpara / para baiṭhā hai*’, both the reference object and the Figure are shown as dependents

[7]postposition

[8]k1 and k7p are tags used for marking syntactico-semantic relations in Hindi Dependency Treebank.

of the sentential head which is the verb:

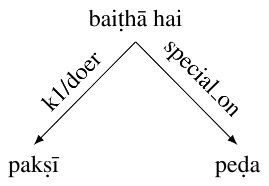


Figure 2: Basic dependency tree for sentence (4)

The semantic label spatial_on indicates that *peḍa* is a reference object for the action 'sitting' whose *kartā* is *pakṣī* (bird). The labels for geometric terms are given in Table-3. Semantic classes in terms of positive and negative values for features are also specified.

The nominal/modificational variant of geometric terms are represented in the form of constructions. Constructions in Construction Grammar (Fillmore, 1988) are linguistic patterns in which some aspect of its form or its meaning cannot be predicted from its component parts. In Indian Grammatical Tradition, it is called *vṛtti*. *Vṛtti* is defined as '*parārthābhidhānam vṛttiḥ*' – an aggregated word-form that gives a sense which is different from the literal sense of its constituents (Joshi et al., 1990; Sharma and Sharma, 1982).

In our construction, geometric terms entail a space which is not lexically expressed but whose geometric relation to the reference object is conveyed by the geometric terms. The linguistic patterns can be represented as a template and the template is assigned a meaning. For example, the following template defines the form-meaning pair for the nominal/modificational usage of the geometric terms:

$$[X]_{\text{ref-obj}} ke\_Term_{\text{spatial/directional-ground}} [ke/vālā][Y]_{\text{loc}} \quad (1)$$

where X and the space denoted by the Term are not in a part-whole relationship.

The subscripts define the meaning and X, ke_Term and Y are variables for physical expressions in 1.

For the sentence (5), the template-1 can be instantiated as follows:

(18) **peḍa**$_{\textbf{ref-obj}}$ **ke_ūpara**$_{\textbf{spatial/directional-ground}}$ **cāṃda**$_{\text{fig}}$ hai

Following are more examples:

(19) **sāmane**$_{\textbf{spatial/directional-ground}}$ / **cāroṃ ora**$_{\textbf{spatial/directional-ground}}$ / **pīche**$_{\textbf{spatial/directional-ground}}$ baḍe baḍe peḍa$_{\text{fig}}$ haiṃ

(20) **ghara**$_{\textbf{ref-obj}}$ **ke sāmane**$_{\textbf{spatial/directional-ground}}$ / **cāroṃ ora**$_{\textbf{spatial/directional-ground}}$ / **pīche**$_{\textbf{spatial/directional-ground}}$ baḍe baḍe peḍa$_{\text{fig}}$ haiṃ

In the above sentences, 'big trees' are the Figure/ *ādheya* and the space denoted by the geometric terms *sāmane / pāsa / pīche* is the Ground / *ādhāra*. In sentences (20) *ghara* is a reference point with respect to which '*ādhā*' has to be interpreted. In (19), the reference point is not explicitly mentioned, and it is from the context the reference point has to be determined. For the above sentences the final part of the template [ke/vālā] [Y]loc is null, and therefore they represent nominal variants. For (21) and (22) below, the construction is used as a modifier and the full template is instantiated as shown below:

(21) **ghara**$_{\textbf{ref-obj}}$ **ke sāmane**$_{\textbf{spatial/directional-ground}}$ / **cāroṃ ora**$_{\textbf{spatial/directional-ground}}$ / **pīche**$_{\textbf{spatial/directional-ground}}$ **ke bagīce**$_{\textbf{loc}}$ **meṃ** baḍe baḍe peḍa$_{\text{fig}}$ haiṃ

(22) **ghara**$_{\textbf{ref-obj}}$ **ke sāmane**$_{\textbf{spatial/directional-ground}}$ / **cāroṃ ora**$_{\textbf{spatial/directional-ground}}$ / **pīche**$_{\textbf{spatial/directional-ground}}$ **vāle bagīce**$_{\textbf{loc}}$ **meṃ** baḍe baḍe peḍa$_{\text{fig}}$ haiṃ

5 Developing a Corpus Search cum Annotation Interface for Geometric Terms

From the above discussion, it has become clear that the semantics of geometric terms are quite chequered. We have done a corpus study to understand how these terms are translated into English. We present here example sentences for 4 geometric terms in Table-4.

From Table-4, we find that *nīce* has been translated into as many as 5 different words while *sāmane* into 4 different words. There is also one case of idiom and one phrasal verb attested in the data. We get the adjectival form of *bāhara* as *bāharī*.

In order to be able to disambiguate the geometric terms so that we get appropriate translations into

Hindi	Bangla	Kannada	Semantic Class	Semantic Label
andara	bhitare	oḷage	(R[9]) Loc: Interior; Con +, (N/M[10]) Loc: Interior; Con -	Spatial inside
bāhara	bāhira, bāire	horage	(R) Loc: Exterior; Con - (N/M) Loc: Exterior; Con -	Spatial outside
āge	āge	muṁde	(R) Loc: Anterior; Dir +	directional ahead
sāmane	sāmane	eduru	(R) Loc: Anterior; Dir +	directional front facing
pīche	pichane	hiṁde	(R) Loc: Posterior; Dir +	directional behind
ūpara	opara, opore, upare	mēle	(R) Loc: Superior; Dir +; Con + (N/M) Loc: Superior; Dir +; Con -	Directional on
nīce	nīce	keḷage	(R) Loc: Interior; Dir +; Con + (N/M) Loc: Interior; Dir +; Con -	Directional under
dāyeṃ	dāine	balakke	(R) Loc: Right_side; Dir +; Con + (N/M) Loc: Right_side; Dir +; Con -	Directional left
bāyeṃ	bāMye	edakke	(R) Loc: Left_side; Dir +; Con + (N/M) Loc: Left_side; Dir +; Con -	Directional right
cāroṃ ora	cāradik, cāridike	suttalu	(R) Loc: Circumferential; Con + (N/M) Loc: Circumferential; Con -	Directional around
bīca	mājhakhāne	madhya	(R) Loc: Medial; Con + (N/M) Loc: Medial; Con -	Spatial between
pāsa	pāśe, kāche	hattira	(R) Approx Spatial Proximity: +; Con + (N/M) Approx Spatial Proximity: +; Con -	Spatial near
dūra	dūre	dūra	(R) Approx Spatial Proximity: -; Con + (N/M) Approx Spatial Proximity: -; Con -	Spatial far

Table 3: Semantic Classification of the geometric terms. The binary feature 'Con' defines the physical 'contact' of the ādheya with the ādhāra; 'Dir' and 'Loc' specify 'direction' and 'location' respectively.

English, we need to understand the context of occurrence of these terms more thoroughly. One way of doing this is to study their usage in corpora, both monolingual and Indian language-English bilingual corpora. To facilitate this study we propose to design an interface for intelligent search of these terms in different contexts. For that purpose, we create a database of existing annotated monolingual corpora as well as a database of parallel corpora. For monolingual corpora, we use POS tags, actual lexical items and syntactic relations as ques for searching the database. When we set out to do that, we find that the geometric terms irrespective of their functional status are annotated as NST in the existing corpora (Jha, 2012). In order to make the search more meaningful, we have integrated an annotation facility to the search interface for annotating the geometric term for relation markers, nouns and modifiers. This will help in future to search these terms for their different functions.

6 Conclusion

The paper has examined the spatio-directional geometric terms and their semantics in a great detail mainly for Hindi and also for Bangla and Kannada. We have observed that even though these geometric terms have some morpho-syntactic differences in these three languages they are very much in alignment in terms of interpretations. This paper is the beginning of an in-depth study of geometric terms of Indian languages. There remains much work to be done in laying out systematically the subtler differences among apparently close terms such as *āge, sāmane; bāhara, sāmane* and so on. For example, we can almost interchangeably say the following two sentences:

[9] 'R': Relational

[10] 'N': Nominal, 'M': Modifier

(23) ghara ke bāhara gāḍī khaḍī hai
house GEN outside car.NOM park.PR.SG
'The car is parked outside the house.'

(24) ghara ke sāmane gāḍī
house.SG GEN in front of car.NOM
khaḍī hai
park.PR.SG
'The car is parked in front of the house.'

But that is not true for the following pair of sentences:

(25) a. meja ke sāmane kursī
table GEN in front of chair.NOM
hai
be.PR.SG
'A chair is kept in front of the table.'

b. *meja ke bāhara kursī
table GEN outside chair.NOM
hai
be.PR.SG
'*A chair is kept outside the table.'

It appears that the semantics of reference objects are playing a role in licensing the geometric terms. This paper draws the conclusion that in Indian languages (at least for those under consideration), spatio-directional geometric terms play three roles: relational, nominal and modificational. We have proposed to design an interface for annotating geometric terms for their different interpretations. The information can be useful for Natural Language Understanding, Natural Language Generation and knowledge rich Machine Translation.

Acknowledgement

We thank Mr Prateek Saxena for his active participation in developing the Annotation cum Search Interface for the geometric terms presented in this paper.

References

Akshar Bharati, Vineet Chaitanya, and Rajeev Sangal. 1995. *Natural Language Processing: a Paninian Perspective*. Prentice-Hall of India New Delhi.

Akshar Bharati, Rajeev Sangal, and Dipti M Sharma. 2007. Ssf: Shakti standard format guide. *Language Technologies Research Centre, International Institute of Information Technology, Hyderabad, India*, pages 1–25.

Akshara Bharati, Dipti Misra Sharma, Samar Husain, Lakshmi Bai, Rafiya Begam, and Rajeev Sangal. 2009. Anncorra: Treebanks for indian languages, guidelines for annotating hindi treebank.

Charles J Fillmore. 1988. The mechanisms of "construction grammar". In *Annual Meeting of the Berkeley Linguistics Society*, volume 14, pages 35–55.

Jean-Michel Fortis and Benjamin Fagard. 2010. Space in language. In *Leipzig school summe*.

Girish Nath Jha. 2012. The tdil program and the indian language corpora initiative. In *Language Resources and Evaluation Conference*.

Shivram Dattatray Joshi, J. A. F. Roodbergen, et al. 1990. *Patañjali's Vyākaraa-Mahābhāya, Samarthāhnika*, volume II. Bhandarkar Oriental Research Institute, Pune.

Yamuna Kachru. 2006. *Hindi*, volume 12. John Benjamins Publishing.

Omkar N Koul. 2008. *Modern Hindi Grammar*. Dunwoody Press Springfield, VA.

Barbara Landau. 2017. Update on "what" and "where" in spatial language: A new division of labor for spatial terms. *Cognitive science*, 41:321–350.

Stephen Levinson, Sérgio Meira, The Language, and Cognition Group. 2003. 'natural concepts' in the spatial topological domain-adpositional meanings in crosslinguistic perspective: An exercise in semantic typology. *Language*, pages 485–516.

Girdhar Sharma and Parmeshwaranand Sharma. 1982. *Laghu Siddhant Kaumudi Part 02*. Motilal Banrassidas Pvt Ltd.

Bhimsen Shastri. 1926. Laghusiddhāntakaumudī bhaimi-vyākhyā. *Delhi, Bhaimi Prakashan*.

KA Subramania Iyer. 1971. *The Vakyapadiiya of Bhartrhari, Chapter Iii, Pt. i*. Poona:[Deccan College Postgraduate and Research Institute.

Leonard Talmy. 1983. How language structures space. In *Spatial orientation*, pages 225–282. Springer.

A Appendix-1

Glosses for the Hindi, Bangla and Kannada examples given in Table-2.

Hindi: Nominal use:-

(26) ladakā pīche se āyā
boy.SG.NOM behind ABL come.3.SG.PT
'The boy came from behind.'

Bangla: Nominal use:-

(27) chele-ṭā pechon theke elo
boy.SG.NOM behind ABL come.3.SG.PT
'The boy came from behind.'

Kannada: Nominal use:-

(28) huḍuga himdininmda bamdanu
boy.SG.NOM behind.ABL come.3.SG.PT
'The boy came from behind.'

Hindi: Relational marker use:-

(29) meja ke nīche
table.SG GEN under
cyūiṃgama cipakī hai
Chewing gum.SG.3.NOM stick.PR.SG.3
'There is a chewing gum stuck under the table.'

Bangla: Relational marker use:-

(30) ṭebil-er nīc-e
table.SG.GEN under
cuing-gum āṭake āche
Chewing gum.SG.3.NOM stick.PR.SG.3
'There is a chewing gum stuck under the table.'

Kannada: Relational marker use:-

(31) ṭebal-ina keḷage
table.SG.GEN under+towards
cyūiṃgama aṁṭide
Chewing gum.SG.3.NOM stick.PR.SG.3
'There is a chewing gum stuck under the table.'

Hindi: Modifier use:-

(32) gara ke pīche (ke) bāgīce
house.SG GEN behind GEN garden.SG
meṃ baḍe baḍe
LOC big.PL.NOM big.PL.NOM
peḍa haiṃ
tree.PL.3.NOM be.PR.3.PL
'There are big trees in the garden which is behind the house.'

Bangla: Modifier use:-

(33) bāriṛ pechone(r)
house.SG.GEN behind.GEN
bāgāne baro baro
garden.SG.LOC big.PL.NOM big.PL.NOM
gāch āche
tree.PL.3.NOM be.PR.3.PL
'There are big trees in the garden which is behind the house.'

Kannada: Modifier use:-

(34) maneya hindina
house.SG.GEN behind.GEN
tōṭadalli doḍḍa
garden.SG.LOC big.PL.NOM
maragaḷive
tree.PL.3.NOM, exist.PR.3.PL
'There exist big trees in the garden which is behind the house.'

down, below, underground, under, underneath	**nīce**	**Tag**
In the same fashion Son river falls **300 metres down.**	usī tarja meṃ sona nadī **300 phīṭa nīce** giratī hai.	N
While going to Kedarnath snow mass are seen slipping **here and there below the feet.**	kedāranātha jāte samaya **pairoṃ ke nīce yatra-tatra** hima rāśi khisakatī dikhāī paḍatī hai.	M
In Bucharest city there are **underground trains** as ell which are called as 'Metro'.	bukhāresta śahara meṃ **jamīna ke nīce calane vālī releṃ** bhī haiṃ, jinheṃ 'meṭro' kahā jātā hai.	N
There are special kind of shoes for skiing, **under which** a long metal ski board is attached.	Skīiṃga ke lie viśeṣa prakāra ke jūte hote haiṃ, **jinake nīce** dhātu kā banā laṃbā skī bleḍa lagā hotā hai.	R
Here **underneath a peepal tree,** Shree Krishna sat in a lugubrious pose.	yahāṃ eka **pīpala ke peḍa ke nīce** śrīkṛṣṇa viṣādamaya mudrā meṃ baiṭhe the.	N
ahead, forward	āge	
The charming valley of Keshkal starts **22 kilometres ahead of Kanker.**	**kāṃkera se 22 kilomīṭara āge** śurū hotī hai keśakāla kī manorama ghāṭī	N
Taking deep breaths making our breath regular, we were **going forward.**	gahare śvāsa bharakara apanī sāṃsoṃ ko niyamita karate hue hama **āge baḍha rahe the.**	N
out, outer, outside	bāhara	
Picnic spots adorned with cedars and rivers and streams are **out of population.**	**ābādī se bāhara** nadīnāloṃ va devadāroṃ se saje pikanika sthala haiṃ.	N
Installed in the Atishay Kshetra the **outer structure** of this temple is extremely grand.	Atiśaya kṣetra meṃ sthāpita isa maṃdira kā **bāharī svarūpa** atyaṃta bhavya hai.	M
Diwan-e-Khas **looks like a one-storey building from outside** but from inside it is double storied.	dīvāna-e-khāsa imārata **bāhara se dekhane meṃ eka maṃjilā pratīta hotī hai** magara aṃdara se domaṃjilā hai.	N
before, in front of, to the fore, out	**sāmane**	
Then Mahalaxmi had **appeared before him** with a lotus in her hand.	taba mahālakṣmī hātha meṃ kamala dhāraṇa kie hue **unake sāmane prakaṭa huī** thīṃ.	R
An extremely attractive pillar is installed **in front of the temple.**	**maṃdira ke sāmane** eka atyaṃta ākarṣaka staṃbha sthāpita hai	N
During the excavation of Vaishali **it came to the fore** that it has had an impressive history.	vaiśālī meṃ milī khudāī meṃ mile avaśeṣoṃ se **yaha bāta sāmane āī** hai ke isakā eka prabhāvaśālī itihāsa rahā hai.	Idiom
At the confluence place of the rivers **which is right before the temple** are beautiful but small falls.	nadiyoṃ ke saṃgama sthala para **jo maṃdira ke ṭhīka sāmane haiṃ** suṃdara kintu choṭā prapāta hai.	N
Although quarrying of the Stupa is still not complete yet its 104 ft high structure has **come out.**	stūpa kā utkhanana-kārya yadyapi abhī pūrā nahīṃ huā hai, tathāpi, isakī 104 phīṭa ūṃcī saṃracanā **sāmane ā cukī hai.**	Phrasal verb
The **part in the front of it** has fallen.	**inakā sāmane kā hissā** gira gayā hai.	M

Table 4: Example sentences from the corpus for 4 geometric terms

Morpheme Boundary Detection & Grammatical Feature Prediction for Gujarati : Dataset & Model

Jatayu Baxi
Dharmsinh Desai University / Nadiad
jatayubaxi.ce@ddu.ac.in

Dr. Brijesh Bhatt
Dharmsinh Desai University / Nadiad
brij.ce@ddu.ac.in

Abstract

Developing Natural Language Processing resources for a low resource language is a challenging but essential task. In this paper, we present a Morphological Analyzer for Gujarati. We have used a Bi-Directional LSTM based approach to perform morpheme boundary detection and grammatical feature tagging. We have created a data set of Gujarati words with lemma and grammatical features. The Bi-LSTM based model of Morph Analyzer discussed in the paper handles the language morphology effectively without the knowledge of any hand-crafted suffix rules. To the best of our knowledge, this is the first dataset and morph analyzer model for the Gujarati language which performs both grammatical feature tagging and morpheme boundary detection tasks.

1 Introduction

As Natural Language Processing is increasingly becoming an active area of research with many important applications, most of the research is focused only on a few languages. Developing NLP tools for under resource languages is an essential task, as it not only opens considerable economic perspectives but also prevents its extinction and foster its expansion (Magueresse et al., 2020)

In this paper we present a morph analyzer for Gujarati. Gujarati is an Indo-Aryan language, spoken mainly in the Gujarat state of India. It is the 26th most widely spoken language with approximately 55 million speakers across the world. We identify various grammatical features of Gujarati morphology and prepare a gold data set of 16527 unique words. We have developed Bi-LSTM based morph analyzer inspired from (Premjith et al., 2018) and (Tkachenko and Sirts, 2018). For the training and evaluation of the morph analysis task, we have built the dataset for Gujarati language in the standard Unimorph format (Kirov et al., 2018). The neural architecture proposed in this paper does not require any language specific rules and captures linguistic characteristics of the language effectively.

The remaining of the paper is organized as follows : Section 2 describes related work. Section 3 describes the dataset details. Section 4 describes proposed approach. section 5 describes experiments and observations. Section 6 describes result analysis from the linguistic perspective. In section 7, we discuss conclusion and future research direction.

2 Related Work

Morphological analysis is the task of analyzing the structure of the morphemes in a word and is generally a prelude to further complex NLP tasks such as parsing, machine translation, semantic analysis.

Existing approaches to build a Morphological Analyzer can be broadly classified as Rule Based approaches and Machine Learning based approaches. Recent breakthroughs in the field of Deep Learning has motivated researchers to apply the neural models to the Morphological Analyzer problem.

Two-level morphology(Koskenniemi, 1984) was the first practical general model in the history of computational linguistics for the analysis of morphologically complex languages. The current C version two-level compiler, called TWOLC, was created at PARC(Beesley and Karttunen, 1992). (Kenneth R. Beesley and Lauri Karttunen, 2003) introduced XFST, a finite state morphology tool. Finite state transducer based approach have been used to develop Morph analyzer for many languages (Beesley, 1998)(Beesley, 2003)(Megerdoomian, 2004).(Kumar et al., 2012) developed a morphological analyzer for Hindi

Proceedings of the 18th International Conference on Natural Language Processing, pages 369–377
Silchar, India. December 16 - 19, 2021. ©2021 NLP Association of India (NLPAI)

using this approach. Following Hindi, morphological generators were developed for other Indian languages e.g. Kannada (Melinamath and Mallikarjunmath, 2011), Oriya (Sahoo, 2003), etc. (Bharati et al., 2002) described another popular approach named Paradigm Based approach for morphological analysis. A Paradigm defines all the word forms that can be generated from given Root along with grammatical feature set.

Apart from the rule based techniques, there have been some efforts to develop machine learning based methods to develop morph analyzer for Indian languages. (Anand Kumar et al., 2010) have defined the morphological analysis problem as classification problem and experimented with various kernel methods to capture non linear relationships of the morphological features using SVMTool. (Srirampur et al., 2015) developed Statistical Morphological Analyzer for the Indian languages using linguistic features.

(Chakrabarty et al., 2016) proposed a Neural lemmatizer for the Bengali language. The proposed lemmatizer makes use of contextual information of the surface word to be lemmatized. (Heigold et al., 2016) investigated character based neural morphological tagger for morphologically rich languages having large tag set. They presented various neural architectures. The work is extended in (Heigold et al., 2017) by experimenting on 14 different languages. (Chakrabarty et al., 2017) introduced Deep Neural Network based method for lemmatization. This method works by identifying a correct edit tree for the word lemma transformation. (Premjith et al., 2018) Developed a Deep Learning approach for detecting morpheme boundaries. They studied agglutinative nature and sandhi splitting process of Malayalam language. (Tkachenko and Sirts, 2018) explore 3 different neural architectures named simple multiclass multilabel model, hierarchical model and sequence model for Morphological tagging. The encoder used for this work is same as the one used in (Lample et al., 2016). (Gupta et al., 2020) evaluated various neural morpholgical taggers for Sanskrit with the focus on the fact that good result for morphological tagging should be achieved without an extensive linguistic knowledge.

There have also been effort to develop unsupervised approaches for morphological tagging task. (J., 2005) described an Automatic Morphological Analyser which can be adopted for different languages. Morphessor (Creutz, 2005) is another widely used tool which performs unsupervised morphological segmentation. (Ak and Yildiz, 2012) and (Narasimhan et al., 2015) have also proposed unsupervised morphological analyzer using Trie based approach and Log linear methods.

To the best of our knowledge, very less work is reported in the area of developing a morph analyzer for the Gujarati language. (Patel et al., 2010) built a stemmer using handcrafted suffix list along with unsupervised learning. (Suba et al., 2011) built Hybrid Inflectional Stemmer and Rule-based Derivational Stemmer. (Baxi et al., 2015) developed rule based lemmatizer for Gujarati by hand crafting of suffix rules. The language independant models such as morfessor can not be used to develop full fledge morph analyzer as they only give morphological segmentation and do not perform morphological feature tagging. The model suggested by (Heigold et al., 2017) can not be directly used as the language specific training data for Gujarati language is not available in penn treebank dataset.

3 Gujarati morphology and Data set Generation

Gujarati is a verb-final language and has a relatively free word order, it is an inflectional language[1]. Words are formed by successfully adding suffixes to the root word in series. When suffixes are attached to the root, several morphophonemic changes take place. In this section, we describe the format and the details about the data set creation for the training and evaluation of morphological analyzer. For the creation of the dataset, we did a survey of available corpus for the Gujarati language. For the morphological analysis, it is preferable to have a POS tagged data, hence we have selected Gujarati Monolingual Text Corpus ILCI-II corpus for the creation of the dataset. The dataset is obtained from TDIL.[2]. We now discuss Gujarati morphology and details of dataset created for Noun, Verb and Adjective POS categories.

3.1 Noun

Gujarati nouns participate in three genders and two numbers. The genders are masculine, feminine

[1]https://en.wikipedia.org/wiki/Gujarati$_{g}rammar$

[2]Technology Development for Indian Languages (TDIL), http://http://tdil-dc.in

and neuter and the numbers are singular and plural. Gujarati nouns also inflect for various cases. Table 1 shows various cases with corresponding case markers. The data set for the noun category contains 6847 number of unique nouns. Along with each noun entry, the corresponding root form, gender, number and case information are marked manually.

Case	Suffix
Nominative	ϕ
Genitive	નો,ની,નું,નાં (*Nō,nī,nuṁ,nāṁ*)
Ergative	એ (*ē*)
Objective/Dative	ને (*nē*)
Ablative	થી (*thī*)
Locative	માં (*māṁ*)

Table 1: Case Markers for Gujarati Noun

3.2 Verb

Gujarati verbs inflect for gender, number, person, tense, aspect and mood features. Table 2 and Table 3 shows example of Gujarati verb with different moods and aspects respectively. The dataset for the verb category contains 6334 inflected verb forms. Each verb form is marked with corresponding root form and corresponding linguistic features.

3.3 Adjective

Gujarati adjectives can be classified in two types based on their nature of inflections. One class of adjectives do not inflect while the other class inflect for gender and number. Table 4 shows example of each category. The dataset for the adjective contains 3346 inflected adjective forms marked with linguistic features type, gender and number.

POS Category	Features	Number of Words
Noun	Gender, Number, Case	6847
Verb	Gender, Number, Tense, Aspect, Person	10128
Adjective	Gender, Number	3346

Table 5: Details about Dataset

4 Proposed Approach

We propose a morphological analyzer for Gujarati which performs morpheme boundary detection and grammatical feature tagging of a given inflected word. Gujarati is a morphologically rich language and manual hand crafting of rules is cumbersome process, hence we propose a deep learning based approach for this problem so that the features of an inflected word can be learned without supplying hand crafted rules. The morpheme boundary segmentation model inspired by (Premjith et al., 2018) and feature tagging module is based on the work reported in (Tkachenko and Sirts, 2018).

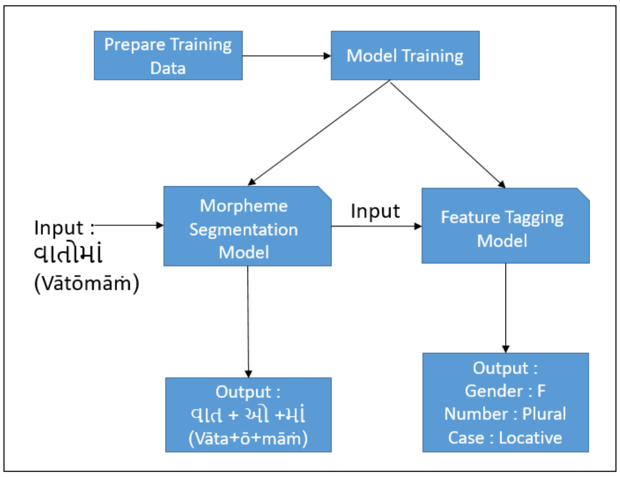

Prepare Training Data
Model Training
Input : વાતોમાં (Vātōmāṁ)
Morpheme Segmentation Model
Input
Feature Tagging Model
Output : વાત + ઓ +માં (Vāta+ō+māṁ)
Output : Gender : F Number : Plural Case : Locative

Figure 1: Block Diagram of the System

Figure 1 shows the overall architecture of the proposed system. We first prepare the training data and perform the model training for both tasks. The input word is passed to both models to get the corresponding morpheme segmentation and linguistic feature tagging outputs. For the preparation of training data, we represent inflected word as binary string and mark "1" in the position of the split character, rest all characters are marked as "0". Figure 2 shows morpheme splitting example. Due to the rich morphological nature of Gujarati language, different word forms can be constructed by attaching various suffixes to the root word. Table 6 shows the sample output obtained for morpheme segmentation task.

Mood	Example	Transliteration	English Translation
Indicative	ધન્વી ચોકલેટ ખાય છે.	*Dhanvī cōkalēṭa khāya chē.*	*Dhanvi is eating a chocolate.*
Imperative	સવારે વહેલો ઊઠજે.	*Savārē vahēlō ūṭhajē.*	*Get up early in the morning..*
Conditional	જો હું ત્યાં હોત, તો હું તમને મદદ કરી શક્યો હોત.	*Jō huṁ tyāṁ hōta, tō huṁ tamanē madada karī śakyō hōta.*	*Had I were there, I would have helped..*
subjunctive	એ અત્યારે દીપક ને ઘેર હોવાનો.	*Ē atyārē dīpaka nē dhēra hōvānō.*	*He must be at Dipak's home right now.*

Table 2: Moods of Gujarati Verb

Mood	Example	Transliteration	English equivilant
Simple	રામ અમદાવાદમાં રહે છે.	*Rāma amadāvādamāṁ rahē chē.*	*Ram lives in Ahmedabad.*
Progressive	રામ અત્યારે પુસ્તક વાંચી રહ્યો છે.	*Rāma atyārē pustaka vāṅcī rahyō chē.*	*Ram is reading a book right now..*
Perfect	રામે પુસ્તક વાંચી લીધું.	*Rāmē pustaka vāṅcī līdhuṁ.*	*Ram has finished reading a book.*
Perfect Progressive	રામ સવારથી પૂજા કરી રહ્યો હતો.	*Rāma savārathī pūjā karī rahyō hatō.*	*Ram was doing pooja since morning.*

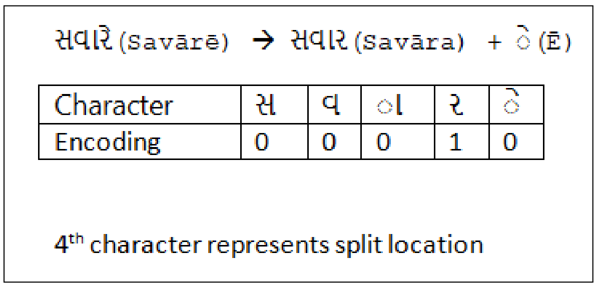
Table 3: Gujarati Verb Aspects

Type of Adjective	Example
Non-Inflected	ઉત્તમ (*Uttama*)
Inflected	સારો, સારી, સારું ,સારા (*sārō,sārī,sāruṁ,sārā*)

Table 4: Gujarati adjective inflection

Inflected Word	Morpheme separation
વાતોમાં(*Vātōmāṁ*)	વાત (*vāta*) + ઓ (*ō*) +માં(*māṁ*)
પત્નીને (*Patnīnē*)	પત્ની(*patnī*) + ને (*nē*)
કરતો હતો(*Karatō hatō*)	કર(*kara*) + તો(*tō*) + હતો(*hatō*)

Table 6: Morpheme Separation Example


સવારે (Savārē) → સવાર (Savāra) + ે (Ē)

Character	સ	વ	ા	ર	ે
Encoding	0	0	0	1	0

4th character represents split location

Figure 2: Morpheme Splitting Example

In the Linguistic Feature Tagging module, we predict morphological features associated with an inflected word. Table 5 describes various features associated with different part of speech categories. We formulate this task as multi class classification problem. We have represented the class labels in a monolithic way such that each unique combination of different features is considered as one class. The number of classes for this task are 36, 198 and 13 for noun, verb and adjective categories respectively.

5 Experiments and Results

We use Keras Python library for the implementation of the system. Our model is sequential model with the first layer as embedding layer followed by a Bi-Directional LSTM layer followed by a dense layer for output prediction. We use Adam optimizer and binary cross entropy and categorical cross entropy loss for morpheme segmentation and feature tagging task respectively. We keep 80:20 ratio for training and testing of the model.Table 7 shows the results obtained for morpheme segmentation task.

# of words in test set	Correctly segmented words	Accuracy
4058	3614	89.05

Table 7: Morpheme boundary detection result - overall

Table 8 shows the results separately for each POS category.

POS Category	# of Words	Correctly predicted words	Accuracy
Noun	1369	1240	90.57 %
Verb	2025	1761	86.96 %
Adjective	669	645	97.49 %

Table 8: Morphme Boundary Detection Results - POS category wise

To study the effect of POS category on the results, we repeat the experiments individually for each POS category. Table 9 shows the result of the morphological tagging task for various POS categories. We observe that system performs very well for morpheme boundary segmentation task across the POS categories.

We also compare the results of the neural morphological analyzer with an existing unsupervised morph analyzer Morphessor. Morphessor(Creutz, 2005) is a family of methods for unsupervised morphological segmentation. The first version of Morfessor, called Morfessor Baseline, was developed by Creutz and Lagus (2002) and its software implementation, Morfessor 1.0 was released by Creutz and Lagus (2005b). We have tested our dataset on morfessor implementation and compared the results of the neural model and the unsupervised model. For the Gujarati language, Morphessor implementation is available in Indic NLP(Kunchukuttan) library which is very popular NLP library for Indian Languages. However, the limitation of Morphessor is that it only performs morpheme segmentation task, whereas our proposed neural morphological analyzer performs both morpheme segmentation and morph feature tagging tasks. Due to this limitation, we are able to compare results for neural and unsupervised morph analyzer for only the morpheme segmentation task. We observe that the neural morphological analyzer outperforms unsupervised model by a large margin.

	Accuracy in %	
POS Category	Neural Model	Unsupervised Model
Noun	90.57	68.27
Verb	86.96	12.95
Adjective	97.49	25.72

Table 10: Accuracy comparison of neural and unsupervised model

6 Result analysis

6.1 Morpheme boundary detection

Using the LSTM based morpheme segmentation module, the system predicts a correct segmentation point for 3613 words out of 4058 total words in the test data set. Table 11 highlights few examples where system identifies correct splitting location for an inflected word:

Even though the morphemes splitting in all above cases are correct, It is observed that the first portion of the split may not be the valid root word every time. Table 12 highlights such examples.

We make an observation that the rules to form a valid root word are different for each word. These rules depend on POS category of the word and other grammatical features. Table 13 summarizes the rules.

POS Category	Accuracy	Precision	Recall	F1-Score
Noun	70.64	0.7	0.68	0.68
Verb	16.18	0.1	0.17	0.12
Adjective	85.85	0.78	0.61	0.68

Table 9: Morphological Feature Tagging Task Result

Segmentation Example
પાત્રમાં (*Pātramāṁ*) → પાત્ર (*Pātra*)+માં (*Māṁ*)
દોડી (*Dōḍī*) → દોડ (*dōḍa*) +◌ી (*ī*)
મગજને (*Magajanē*) → મગજ (*magaja*)+ને (*nē*)
દેખાશે (*Dēkhāśē*)→ દેખા (*dēkhā*)+ શે (*śē*)
યંત્રો (*Yantrō*) → યંત્ર (*yantra*)+◌ો (*ō*)
વ્યાજના (*Vyājanā*)→ વ્યાજ (*Vyāja*) + ના (*nā*)
છોકરા (*Chōkarā*) → છોકર (*chōkara*)+◌ા (*ā*)
ધંધાનું (*Dhandhānuṁ*)→ ધંધ(*dhandha*) + ◌ાનું(*ānuṁ*)
ઈશારા(*Īśārā*)→ ઈશાર (*iśāra*) +◌ા(*ā*)

Table 11: Segmentation Examples

POS Category	Other Features	Rule
Noun / Adjective	Gender = Male	Attach Suffix - ◌ો(*Ō*)
Noun / Adjective	Gender = Female	Attach Suffix - ◌ી(*ī*)
Verb	-	Attach Suffix ◌ું(*uṁ*)

Table 13: Rules to form correct root word

We supply POS category of the word as an input to the system and obtain the grammatical features using morph tagging module. Using this information, accuracy of the morpheme boundary detection task can be further enhanced.

It is also observed that due to ambiguities in the word formation rules, in some cases, the system is not able to identify correct segmentation. For example, words વિદેશો (Vidēśō) is spitted correctly as વિદેશ +◌ો (Vidēśa + ō) and word જબરો (Jabarō) is spitted correctly as જબર +◌ો(Jabara + ō) . System tries to split the words ખુલાસો(Khulāsō) and કિનારો(Kinārō) using similar method leading to incorrect outputs ખુલાસ(Khulāsa) and કિનાર(Kināra). The issue here is that system considers ◌ો (Ō) as the suffix but in some words◌ો(Ō) is part as the root word not as a suffix. The similar issue is observed in many other inflected words ending with suffix ◌ી(Ī) as highlighted in the table 14

We also observe that the system does not produce correct segmentation in some cases where multiple suffixes are attached. For example, the correct segmentation of the word કારખાનાનું(*Kārakhānānuṁ*) is કારખાના(Kārakhāna)+◌ા(Ā)+નું(*Nuṁ*) but the system does not identify any segmentation in the given word.

6.2 Grammatical feature prediction task

In this section, we do the result analysis of the grammatical feature prediction task from the linguistic perspective. We perform this analysis individually for each part of the speech category.

Noun:

For Noun, we consider gender, number and case as morphological features. The model is trained in such a way that based on the inflections that a word takes, it predicts corresponding grammatical features. For most of the cases we have good correlation between suffix and grammatical features, but in some cases the correlation does not hold. Due to these exceptions, sometimes there is an error in the feature prediction task. Consider two noun examples બજારો(Bajārō) and ડાયરો(Ḍāyarō). Both words take similar suffix but in word બજારો(Bajārō), the suffix indicates

Inflected word	Root morpheme detected by the system	Actual root word
દેખાશે (*Dēkhāśē*)→ દેખા (*dēkhā*)+ શે (*śē*)	દેખા (*dēkhā*)	દેખા(*dēkhā*) + વું(*vu ṁ*) → દેખાવું(*Dēkhāvu ṁ*)
છોકરા (*Chōkarā*) → છોકર (*chōkara*)+◌ા (*ā*)	છોકર(*chōkara*)	છોકર (*chōkara*)+◌ુ(*U*) +◌ં(*ṁ*) → છોકરું(*Chōkaru ṁ*)
ધંધાનું (*Dhandhānu ṁ*)→ ધંધ(*dhandha*) + ◌ાનું(*ānu ṁ*)	ધંધ (*dhandha*)	ધંધ(*dhandha*) + ◌ો(*Ō*) →ધંધો(*dhandhō*)
ઈશારા(*Īśārā*)→ ઈશાર (*iśāra*) +◌ા(*ā*)	ઈશાર(*iśāra*)	ઈશાર(*iśāra*) + ◌ો(*Ō*) → ઈશારો (*Īśārō*)

Table 12: Examples of incorrect root identification

Segmentation	Remark
છોકરી(*Chōkarī*) → છોકર(*Chōkara*) + ◌ી(*I*)	Correct Segmentation
ગણતરી(*gaṇatarī*) → ગણતર(*gaṇatara*) + ◌ી(*I*)	Incorrect Segmentation
શ્રેણી(*śrēṇī*) → શ્રેણ(*śrēṇa*) + ◌ી(*I*)	Incorrect Segmentation

Table 14: Segmentation Analysis

plural marker but for the word ડાયરો(Ḍāyarō), the suffix is part of the word itself and the word is not plural. Similarly the word ઘટના(Ghaṭanā) is tagged with genitive case marker due to ના(Nā) attachment but actually the suffix is part of the word.

Verb:
For the verb category, we consider gender, number, person, tense and aspect features. Due to different combinations of features, we get total 198 classes for tag prediction task. The accuracy of the prediction task for verb is poor due to large number of classes. It is also observed that for different combinations of the features, same verb form exists which makes classification task more difficult. Table 15 highlights such examples:

A possible solution to address the above issue is to look at the input at the sentence level rather than word level. When the sentence level input is taken, verb features becomes clear and unambiguous. For example, with reference to the examples 1 and 2 from the above table, by looking at only રમતો હતો(Ramatō hatō), the person feature is not clear but when we look at the whole sentence : રામ રમતો હતો(Rāma ramatō hatō), the person feature becomes unambiguous (person = 3rd). Similarly, by looking at only રમતી હતી(Ramatī hatī), the number feature is not clear but when we look at the whole sentence:છોકરીઓ રમતી હતી(Chōkarīō ramatī hatī), the Number feature becomes unambiguous (Number=PL) .

Adjective
We consider the type of an adjective, gender and number as features for morph feature tagging of an adjective. Consider the adjective અજ્ઞાની(Ajñānī). As per the language specification, this adjective does not inflect with gender and number but by looking at ◌ી(Ī) suffix, the system predicts it as inflecting type of adjective with female gender.

To summarize, we observe that linguistic issues such as stem to root word generation, attachment of multiple suffixes and ambiguity in suffix rules affects the performance of the system.

7 Conclusion and Future Scope

In this paper we have proposed a Bi-LSTM based morphological analyzer for the Gujarati language. We have prepared the dataset and evaluated the proposed system. The system effectively performs morpheme boundary detection and morphological feature tagging tasks. With the proposed system, morphological analysis of unknown inflected word can be performed without the knowledge of linguistic rules. We have done result analysis from the linguistic perspective. We also conclude that the proposed model performs better than the existing unsupervised model.

In future, we aim to expand the dataset, and implement other neural architectures such as seq2seq model. We also aim to study sentence level dependency for morphological analysis.

Sr No	Features	Verb Form
1	Gender= Male , Number=SG, Person=1st , Tense=Past, Aspect=Progressive	રમતો હતો(*Ramatō hatō*)
2	Gender= Male , Number=SG, Person=3rd , Tense=Past, Aspect=Progressive	રમતો હતો(*Ramatō hatō*)
3	Gender=Female, Number=SG, Person=3rd, Tense=Past, Aspect= Progressive	રમતી હતી(*ramatī hatī*)
4	Gender=Female, Number=PL, Person=3rd, Tense=Past, Aspect= Progressive	રમતી હતી(*ramatī hatī*)

Table 15: Ambiguity in verb form generation

Acknowledgement

The PARAM Shavak HPC computer facility is used for some of our experiments. We are grateful to the Gujarat Council of Science and Technology (GUJCOST) for providing this facility to the institution so that deep learning studies are being carried out effectively.

References

Koray Ak and Olcay Taner Yildiz. 2012. Unsupervised Morphological Analysis Using Tries. *Computer and Information Sciences II*, pages 69–75.

M Anand Kumar, V Dhanalakshmi, K.p Soman, and S Rajendran. 2010. A Sequence Labeling Approach to Morphological Analyzer for Tamil Language. *International Journal on Computer Science and Engineering*, 02(06):1944–1951.

Jatayu Baxi, Pooja Patel, and Brijesh Bhatt. 2015. Morphological Analyzer for Gujarati using Paradigm based approach with Knowledge based and Statistical Methods. *Proceedings of the 12th International Conference on Natural Language Processing*, (December):178–182.

Ken Beesley. 2003. Finite-State Morphological Analysis and Generation for Aymara: Project Report. *Proceedings of the Workshop of Finite-State Methods in Natural Language Processing, 10th Conference of the European Chapter of the Association for Computational Linguistics.*, 5:2–5.

Kenneth R. Beesley. 1998. Arabic morphology using only finite-state operations. *Proceedings of the Workshop on Computational Approaches to Semitic languages. Association for Computational Linguistics*, page 50.

Kenneth R. Beesley and Lauri Karttunen. 1992. Two-Level Rule Compiler. *Technical Report. Xerox Palo Alto Research Center. Palo Alto, California.*

Akshar Bharati, Vinit Chaitanya, Rajeev Sangal, and Brendan Gillon. 2002. *Natural Language Processing: A Paninian Perspective*. prentice hall of india.

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, Portorož, Slovenia. European Language Resources Association (ELRA).

Abhisek Chakrabarty, Onkar Arun Pandit, and Utpal Garain. 2017. Context sensitive lemmatization using two successive bidirectional gated recurrent networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1481–1491, Vancouver, Canada. Association for Computational Linguistics.

K. Creutz, M. Lagus. 2005. Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. *Computer and Information Science, Helsinki University of Technology.*

Ashim Gupta, Amrith Krishna, Pawan Goyal, and Oliver Hellwig. 2020. Evaluating neural morphological taggers for Sanskrit. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 198–203, Online. Association for Computational Linguistics.

Georg Heigold, Guenter Neumann, and Josef van Genabith. 2017. An extensive empirical evaluation of character-based morphological tagging for 14 languages. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 505–513, Valencia, Spain. Association for Computational Linguistics.

Georg Heigold, Guenter Neumann, and Josef van Genabith. 2016. Neural Morphological Tagging from Characters for Morphologically Rich Languages. *arXiv e-prints*, page arXiv:1606.06640.

Goldsmith J. 2005. Unsupervised learning of the morphology of a natural language. *Computational Linguistics 27(2)*, pages 153–198.

Kenneth R. Beesley and Lauri Karttunen. 2003. Finite-State Morphology. (January).

Christo Kirov, Ryan Cotterell, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sabrina J. Mielke, Arya McCarthy, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2018. UniMorph 2.0: Universal morphology. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Kimmo Koskenniemi. 1984. A general computational model for word-form recognition and production. (July):178–181.

Deepak Kumar, Manjeet Singh, and Seema Shukla. 2012. FST Based Morphological Analyzer for Hindi Language. *arXiv preprint arXiv:1207.5409*.

Anoop Kunchukuttan. *Indic NLP Python Library*. `https://anoopkunchukuttan.github.io/indic_nlp_library/`.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Alexandre Magueresse, Vincent Carles, and Evan Heetderks. 2020. Low-resource Languages: A Review of Past Work and Future Challenges. *arXiv e-prints*, page arXiv:2006.07264.

Karine Megerdoomian. 2004. Finite-state morphological analysis of Persian. *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages. Association for Computational Linguistics*, page 35.

Bhuvaneshwari C. Melinamath and A. G. Mallikarjunmath. 2011. A morphological generator for Kannada based on finite state transducers. *ICECT 2011 - 2011 3rd International Conference on Electronics Computer Technology*, 1:312–316.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. 2015. An Unsupervised Method for Uncovering Morphological Chains. *Transactions of the Association for Computational Linguistics*, 3:157–167.

Pratikkumar Patel, Kashyap Popat, and Pushpak Bhattacharyya. 2010. Hybrid Stemmer for Gujarati. In *Proceedings of the 1st Workshop on South and Southeast Asian Natural Language Processing (WS-SANLP)*, August, pages 51–55, Beijing.

B. Premjith, K. P. Soman, and M. Anand Kumar. 2018. A deep learning approach for Malayalam morphological analysis at character level. *Procedia Computer Science*, 132:47–54.

Kalyanamalini Sahoo. 2003. Oriya nominal forms: A finite state processing. *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, 3:730–734.

Saikrishna Srirampur, Ravi Chandibhamar, and Radhika Mamidi. 2015. Statistical Morph Analyzer (SMA++) for Indian Languages. *Proceedings ofthe First Workshop on Applying NLP Tools to Similar Languages, Varieties and Dialects*, pages 103–109.

Kartik Suba, Dipti Jiandani, and Pushpak Bhattacharyya. 2011. Hybrid Inflectional Stemmer and Rule-based Derivational Stemmer for Gujarati. *Proceedings of the 2nd Workshop on South and Southeast Asian Natural Language Processing (WS-SANLP)*, pages 1–8.

Alexander Tkachenko and Kairit Sirts. 2018. Modeling composite labels for neural morphological tagging. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 368–379, Brussels, Belgium. Association for Computational Linguistics.

Auditing Keyword Queries Over Text Documents

Apparreddy Bharath Kumar Reddy[1] * **Sailaja Rajanala**[2] **Manish Singh**[3]
Department of Computer Science and Engineering
Indian Institute of Technology Hyderabad
{[1]cs15mtech11001, [2]cs15resch11009 }@iith.ac.in,[3]msingh@cse.iith.ac.in

Abstract

Data security and privacy is an issue of growing importance in the healthcare domain. In this paper, we present an auditing system to detect privacy violations for unstructured text documents such as healthcare records. Given a sensitive document, we present an anomaly detection algorithm that can find the top-k suspicious keyword queries that may have accessed the sensitive document. Since unstructured healthcare data, such as medical reports and query logs, are not easily available for public research, in this paper, we show how one can use the publicly available DBLP data to create an equivalent healthcare data and query log, which can then be used for experimental evaluation.

1 Introduction

Large business enterprises, hospitals, etc., maintain a large amount of digital information in the form of structured, semi-structured, and unstructured data. With growing concern among users regarding the privacy of their data, such organizations are required to design a robust data management system. Thus, the goal of DBMS has expanded, to include additional features, such as enforcing data privacy and security (Robling Denning, 1982; Denning et al., 1979), in addition to the primary goal of easy and efficient retrieval.

Lot of research has been done (Duncan and Mukherjee, 1991; Jajodia and Meadows, 1995; Brodsky et al., 2000) to prevent and detect privacy violation for structured data (e.g. SQL) and semi-structured data (e.g. XML) (Byun et al., 2005). However, to the best of our knowledge, there is no existing work that detects privacy violations in access to unstructured text documents using keyword queries. Detecting privacy violations for text documents has not been explored much because it is difficult to audit keyword queries for text data, which is explained later in this section. Organizations tend to maintain their sensitive data in a structured or semi-structured format. However, just as the strength of a chain is equal to its weakest link; similarly, an organization with very secured access to structured and semi-structured can still face privacy violations due to its unsecured unstructured data repositories.

Example *Alice has undergone breast cancer medical treatment in HealthCo Hospital. A few weeks after she returned from the hospital, she started getting advertisements on natural products to treat breast cancer. She blamed HealthCo for disclosing her sensitive disease data to outsiders. HealthCo has a strong security system that will not allow outsiders to directly access Alice's information. HealthCo has to prove that either nobody has misused Alice's private information or find the employees from HealthCo whose access to the information seems suspicious.*

One can use access control policies to secure access to sensitive documents so that only authorized users can access those documents. However, this can restrict access to crucial information at times of emergency. For example, in a hospital, if we create a strict access control policy over medical reports, then it may lead to the inaccessibility of information during crucial hours. An auditing system can help in such scenarios by allowing a relaxed access control policy, and then providing means to detect privacy violations through auditing.

The auditing models that have been proposed for structured or semi-structured data (Agrawal et al., 2004; Bottcher and Steinmetz, 2006; Miklau and Suciu, 2007; Motwani et al., 2008) cannot be used for text documents due to the difference in the query model. Structured and semi-structured data

*Work done during graduate course at Indian Institute of Technology Hyderabad is often accessed using precise query languages, such as SQL or XQuery, which returns the result using the boolean retrieval model. In these query models, there is no notion of ordering among returned tuples or elements. A query is marked suspicious if its result contains any sensitive tuple, or if a sensitive tuple can be inferred from the query result. The auditing techniques proposed for these types of data do not have a notion of the degree of suspiciousness for a query.

Text documents are commonly accessed using keyword queries, which are not precise. And thus, neither the query nor the result indicates what information the user was looking for. Each query returns a long list of documents ordered by some relevant measure, and in most cases, users may look at only the top few results. The major success of IR is due to the ordered nature of its result set. Thus, rather than just returning a long list of queries that had the particular sensitive document in its result, we need to define a suspiciousness order for the queries, using various factors such as the rank of the document in the result, the relevance of the document to the query, access anomalousness, etc.

Auditing is common in the healthcare domain as it involves sensitive patient data. To our knowledge, there is only one publicly available healthcare dataset[1] of medical reports. The dataset has been reported for around 200 hundred patients and has no associated query log. Since there is no existing big publicly available dataset from healthcare that can be used to evaluate auditing systems for unstructured data. In this paper, one of our main contributions is to model healthcare data using DBLP data, which is a large dataset that contains bibliographic information about computer science journals and proceedings. In the medical domain, access is anomalous if someone accesses sensitive information that one is normally not required to access. Doctors or nurses are allowed to access sensitive information based on their needs. As different staff have different roles in a hospital and the role determines whether an access is anomalous or not, we show how one can model such roles and accesses using bibliographic data.

This paper is organized as follows: Section 2 contains related work. Section 3 discusses the auditing model and system architecture. Section 4 discusses proposed algorithms. Section 6 presents evaluation. Finally, Section 7 concludes the paper with some directions for future work.

[1]https://webscope.sandbox.yahoo.com/catalog.php

2 Related Work

Inspired by the Hippocratic Oath[2], the Hippocratic databases were proposed by (Agrawal et al., 2002) that impose data privacy and security protocols on the data.

(Agrawal et al., 2004) address the problem of detecting privacy violations in the case of relational databases. In this work, the authors proposed a framework for detecting whether or not a relational database is adhering to data disclosure policies. Users specify sensitive information in the form of audit expressions. The audit component takes audit expressions and returns all the queries that accessed sensitive data during execution. (Motwani et al., 2008) also study the problem of auditing SQL queries. Given a forbidden view of a relational database, which should be confidential, and a batch of SQL queries posted over the database. It determines whether a query batch is suspicious or not with respect to the forbidden view. (Stoffel and Studer, 2005) use the database views are used to make the decisions on privacy violations. The work proposes to solve the problem of data privacy by looking for the data leak from a view of the database.

(Bottcher and Steinmetz, 2006) proposed an audit system for sensitive XML databases and XPath query language that uses an audit query to describe sensitive information. It also discusses privacy violations in the case of an attacker who submits multiple queries. (Bertino et al., 2001; Bertino and Ferrari, 2002; Damiani et al., 2000; Kudo and Hada, 2000) propose access control approaches for XML data sources ranging from policies to fragments of XML databases.

Another interesting approach towards data privacy comes from inference methods by (Farkas and Jajodia, 2002). Inference based methods stem from the fact that the access control methods can only block direct access to the sensitive, while there still exists ways of inferring the sensitive data through indirect means. We propose an auditing model for unstructured text documents. Our work is motivated by various works done in the structured and semi-structured database to ensure data privacy.

[2]An oath was historically taken by physicians stating their obligations and proper conduct.

3 Text Auditing System

In this section we walk-through the various components of the text auditing system and further illustrate the working example introduced in Section 1.

Figure 1 presents the skeletal view of the auditing system. Users access the unstructured data repository using keyword queries, where the documents are ranked using any IR ranking algorithm. Given a query, we store the following information in its query log: query ID, query, query timestamp, the ID of the user who issued the query, and the IDs of top-n documents returned for the query. Using the query log we maintain an access index, which greatly improves the performance of our auditing system. Access index is an inverted index from document to queries. For each document ID, we have a posting list that contains the IDs of all the queries that contain the document in their top-n results

A user can submit an audit request $audit(d)$, where d is the sensitive document. The access index is used to find all the queries that had the document d in their top-n result. We call this query set the candidate queries. Here, we assume that users will not be interested in results appearing below the top-n. Henceforth, we first find the candidate set of suspicious queries using the stored query log. A query from the candidate set is termed anomalous if its score crosses a certain threshold. In Section 4, we shall decode how a simple suspicious candidate query transcends into an anomalous one.

Let us carry forward the example in Section 1 in-order to better understand the audit scenario. The Table 1, contains five candidate queries that had Alice's report on breast cancer in their top-n results. For each query, we compute two scores: Suspiciousness score (SScore) and Anomalousness score (AScore). The SScore is a measure of how relevant the query is to the audit document, which also indicates how likely the user has seen the sensitive information. AScore is a measure of how unlikely the query is from the user. The unlikely queries are considered anomalous.

In Table 1, Query 4 has the highest SScore since Alice's blood report contains the information about her breast cancer. Although this query is suspicious, it is not anomalous because Lucy is a Nurse in the Oncology department, and it is normal for her to access such reports. The same argument holds true for Query 1.

Query 2, 3 and 5 are from employees of the Gynaecology and Cardiology department, who are not typically expected to access breast cancer-related information. Although Query 3 and 5 are not asking for any information directly related to Alice, they still have high SScore because Alice's audit document has high relevance for these queries. Both Barbara and Chris might be trying to access the information indirectly. The former is trying to get all breast cancer patients from a particular location, and the latter knows that those who have +ve estrogen receptors are likely to have breast cancer. Although Query 2 is accessing some information about Alice, it has low SScore because it has low relevance to Alice's breast cancer data.

If we want the top-2 anomalous queries, then Query 5 and Query 3 will be returned by our system as they have high suspicious scores and are also anomalous. Given the imprecise nature of keyword queries and the lack of user's background information, such as Role, Department, etc., it is difficult to compute SScore or AScore for queries. In general, it is difficult to get the background information of users as it may not be available or one user may have multiple roles. In this paper, we present an algorithm that computes the AScore without having the prior background information of users who issued the query.

4 Algorithms

In Section 3, we explained how we use the access index to find all queries that had the audit document in the top-n query result. We call them as the candidate suspicious queries. We now discuss the elements that make up the SScore and AScore for each candidate query.

4.1 Suspiciousness Score

The Suspiciousness score (SScore) builds on the relevance of the audit document to a query. A precise query is very likely to pull the relevant document on the top. Such queries will land higher SScores concerning the sensitive document. Next, we study a few popular choices for SScore.

Query Relevance: An IR ranking function returns the documents in decreasing order of similarity score to the query q. We call this similarity score as IRScore.

$$IRScore(q, d) = similarity(R, q, d) \qquad (1)$$

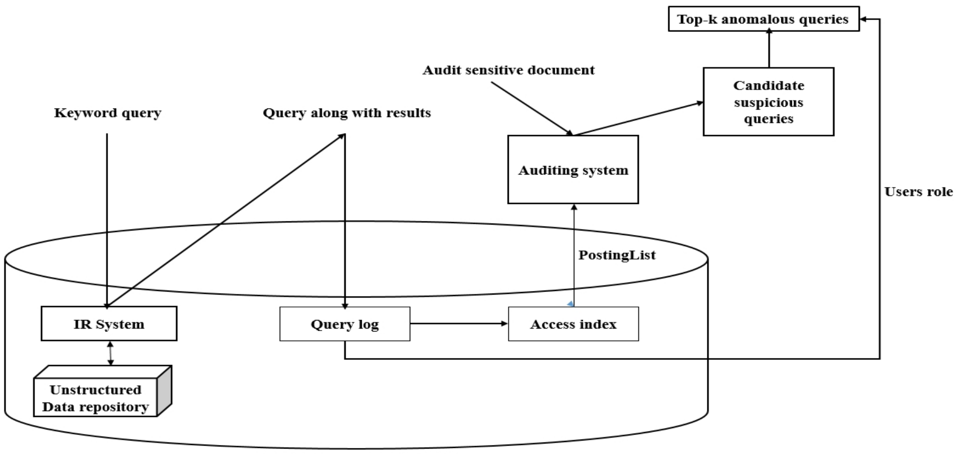


Figure 1: Architecture of text auditing system

Query ID	Username	Department	Role	Keyword query	SScore	AScore
1	Bob	Oncology	Doctor	Non-invasive breast cancer	0.4	No
2	Carol	Gynecology	Doctor	Alice urine report	0.15	Yes
3	Barbara	Gynecology	Nurse	Breast cancer Ann Arbor	0.37	Yes
4	Lucy	Oncology	Nurse	Blood reports of Alice	0.8	No
5	Chris	Cardiology	Doctor	Reports with +ve estrogen receptor	0.45	Yes

Table 1: Sample queries that have accessed Alice's breast cancer report

where R is the IR ranking function, q is the input query and d is the document. If the query has a high similarity with the document, then it indicates that the user who issued the query may be interested in the document.

Document Rank: In IR, results are shown in ranked order of decreasing IRScore i.e., Eq. 1 . The position of a document in the ranked list determines its ease of access. The IRScore can only estimate the significance of a document w.r.t a query. But the relevance of a document is relative to all other documents in the results set. A document catches the attention of the user when it falls within a certain percentile of ranks. In case of a generic query, a majority of the documents have high IRScore. Nevertheless, the user may not look at a document that has high IRScore but does not appear in the top 20-30 documents. On the other hand, indirect queries are vague thus all the documents in the result set have low IRScores. As a result the user can still access the document if it appears in the topmost suggestions (e.g. Query 3 in Table 1) even though it possesses low IRScore.

Therefore, we define IRRank to take into account the rank of result documents in the retrieval system. One can use either the rank of the document or the page number in which the document appears. From empirical evaluation, we observed that considering page number is better than document rank. We thus define IRRank as:

$$IRRank(q,d) = e^{-\lfloor \frac{r}{N} \rfloor} \tag{2}$$

where r is the document rank and N is the number of documents shown per page. By using this function, all the documents that appear on page i have IRRank score of e^{-i}. In general, users only look at the top few pages and the likelihood of their seeing two documents that appear on the same page is equal, so we chose this pagewise exponentially decreasing scoring function.

Click Log: Users do not click on arbitrary links but make selective choices. The click log contains information about the query, the ranked list of documents presented to the user, and the set of links the user clicked (Joachims, 2002). Although a user may not click a sensitive document, by looking at the document snippet in the result set or just its presence in the result set may reveal its sensitive information. Click log information is not definitive of the suspiciousness since it does not record the cases where the user hovers over the document without clicking on it. As a result, it is not used in our paper.

Time Spent by the User: A user spending lot of time on a particular page indicates that he is interested in the documents present on that page, thus his access should be more suspicious to documents that appear on that page. If we know the amount of the time user spent on each page, we can include it in IRRank by defining it as follows:

$$IRRank(q, d, t_i) = e^{-\lfloor \frac{(\frac{r}{N})}{t_i} \rfloor} \quad (3)$$

where t_i is the time (in minutes) that is spent by the user on a page $\lfloor \frac{r}{N} \rfloor$. The IRRank of all the documents in a page increases if the user spends more time on that result page.

SScore of a query for a given sensitive document d is defined as the product of the IRScore and the IRRank of the query for the given document. The set of Candidate Suspicious Queries (CSQ) can be defined as the top-m queries with respect to SScore, or all the queries with SScore more than some given threshold. We use a threshold in this paper.

4.2 Anomalousness Score

In this section, we compute the anomalousness score for the Candidate Suspicious Queries. We compute the anomalousness score using the following two steps: (a) For each user find the topics of interest; (b) Determine anomalous score for each query that accessed the document by computing how anomalous the query is to the topics of interest of the user who issued the query. We determine the anomalousness of the user's topics of interest by comparing them with the topics of interest of other users who have also accessed the document. We explain these two steps below.

Topics of Interest: To find a user's topics of interest, we consider all the user's queries. We then take the union of top-20 result documents for each user's query and denote it as the set S_u. The user's topics of interest are then computed using three topic modeling algorithms, namely TF-IDF, LDA (Blei et al., 2003) and TNG (Wang et al., 2007), on S_u. The TF-IDF representation is the most straightforward approach computed by selecting the top-k words with the highest TF-IDF score.

LDA is a general probabilistic topic modeling algorithm. It is extensively used to determine important topics and terms from a collection of documents. We apply LDA on S_u to get the user's topic of interest. LDA considers each document as a mixture of topics and places frequently co-occurring terms under the same topic with high probabilities. It computes the document-topic distribution (θ) and term-topic distribution (ϕ), which signify the importance of topics in a document and the importance of terms in a topic respectively. The document-topic distribution (θ) is defined as follows:

$$\frac{N_{dj}^{DT} + \alpha}{\sum_{k=1}^{T} N_{dk}^{DT} + T\alpha} \quad (4)$$

where N_{dj}^{DT} is the number of times a term appears in document d that has been assigned to topic j. D and T stand for the document, topic respectively. α is a smoothing constant. Similarly, term-topic distribution (ϕ) is computed as follows:

$$\frac{N_{ij}^{WT} + \beta}{\sum_{k=1}^{W} N_{kj}^{WT} + T\beta} \quad (5)$$

where N_{ij}^{WT} is the number of occurrences of a word i that has been assigned to topic j. W and T represent the terms, topics respectively. β is a smoothing constant. LDA generates the topics from S_u, and each of these topics contains unigram words (terms).

The above methods do not generate topic phrases. Phrases are important to convey a specific meaning. The meaning of 'natural language processing cannot be completely captured by any of the individual words of this phrase. To overcome this problem, we use TNG, which generates topical collocations as well as better unigram words. We use TNG to generate the topic of interest of users from the document set S_u. Similar to LDA, we generate top-m topics.

Query Anomaly: To determine query anomaly, we take all the queries that had the sensitive document d in their top-n result, say $n = 30$. We use these queries and the query log to find the set of all the users $\mathcal{U}_d$ who had the document d in their top-n result. We say a query is anomalous if the user who issued the query has a topic interest that is very different compared to the document's topic.

However, we cannot directly compute the topics from a document because a document has very limited information. Topic modeling algorithms generate good topics only if the corpus has a large amount of data. In a small single document, each term would be present only a few times, so we cannot determine the term importance directly from the document. To address this challenge, we do not directly compare the topics of interest of users with the topics in the sensitive document. We use an indirect approach, where we look at all the users who have accessed the document, and from those users, we find users whose topics of interest are anomalous. Our problem can be formally defined as follows:

Problem *Given a set of users* $\mathcal{U}_d = \{X_1, X_2, ...X_m\}$ *who had the document* d *in their top-*n *result. The access anomaly score of user* X_i *is equal to his average distance from his* k*-nearest neighbors in* $\mathcal{U}_d - X_i$.

We use nearest neighbors to define the anomalous score. If a user has topics of interest that are very different from other similar users who have also accessed the document, then that user would get a high average distance score.

Given two users X_i and X_j, we define their similarity as the cosine similarity of their topics of interest. For topics using TF-IDF, we can directly compute the cosine similarity between topics of interest vector by taking each topic as a term and the TF-IDF score as the term importance. However, we cannot use cosine similarity for the topic distribution obtained using probabilistic topic modeling algorithms, such as LDA or TNG. These algorithms will generate a set of topics with document-topic distribution probability (θ), and for each topic, they will generate a set of terms with term-topic distribution (ϕ). The same term may be present in multiple topics with a different degree of term importance. To use cosine similarity we need to have a document-topic vector that has one importance score per term, where the topic could be a unigram or phrase term. To compute this type of vector, for each term we compute its weight by multiplying the document-topic distribution probability (θ) with term-topic distribution (ϕ). The probability θ indicates the importance of the topic and the probability ϕ indicates the importance of the term in that topic. If a term is present in multiple topics, then the score we assign to the term is the maximum value of the product of the corresponding θ and ϕ values.

5 Modeling Healthcare Data using DBLP

This section shows how to create equivalent healthcare data using the DBLP dataset because such healthcare data is not publicly available for research. We first describe our dataset and then explain the modeling of roles and specializations, generate query logs, and find ground-truth anomalous queries. DBLP[3] is a bibliographic dataset containing information of 3.66 million publications from Computer Science. Each publication in the dataset has information such as title, conference name, the name of authors, publication year, etc. We removed publications that do not contain the name of authors, abstract, or conference name from the dataset. Our processed dataset contained 183232 publications.

5.1 Modeling Specializations and Roles

In healthcare, we consider access anomalous if an employee accesses sensitive information that he usually is not required to see. Since in hospital, each employee has his specialization(s) and the department he belongs to, we can compare the accesses of the particular employee with other employees with similar roles and departments to determine whether the access is anomalous or not.

Although this information is not directly available in DBLP data, we use authors' publications to find their area of research. If an author publishes or looks for a very different paper from his research area, then we consider such papers anomalous. For example, Prof. H. V. Jagadish[4] is a well-known researcher in the area of Databases and Data Mining. However, one of his papers: "Hui Jin, H. V. Jagadish: Indexing Hidden Markov Models for Music Retrieval. ISMIR 2002" seems like an outlier given most of his other papers are in Databases.

Since DBLP does not have the research interest of authors, we combine it with WikiCFP[5] to get the research interest of authors. WikiCFP is a website that advertises calls for papers of international workshops, conferences, and journals. While posting CFP for a conference, one can tag the conference with one or more research areas. We crawled this information from WikiCFP to get the research interest of authors.

For each author, we consider all his publications in the DBLP dataset. We use the conference name

[3]https://www.aminer.cn/dblp_citation

[4]https://web.eecs.umich.edu/ jag/

[5]http://www.wikicfp.com/cfp/

Research Area 1	**Research Area 2**	**Jaccard Similarity**
Health Informatics	E-health	1
Formal Methods	Verification	0.67
Parallel Computing	HPC	0.41
Education	E-learning	0.36
Machine learning	Verification	0.05

Table 2: Jaccard similarity between research areas

Conference Name	**Areas of the conference**
VLDB	Databases
ICVS	Computer Vision, **Pattern Recognition**, **Image Processing**
CIKM	Web, Information Management, Data Mining, Information Retrieval, Text Mining, Databases, **Knowledge Engineering**, **Knowledge Management**, **Database**
KDD	Web, Data Mining, Machine Learning, Information Retrieval, Databases, Knowledge Discovery, Data Science, Big Data, **Knowledge Engineering**

Table 3: A sample of conferences and their research areas as crawled from WikiCFP. Highlighted in bold are areas extended using Jaccard similarity.

in the WikiCFP dataset to get the area(s) of the particular publication. Since there are thousands of category labels in WikiCFP, the CFP poster may not label a conference with all the possible labels. For example, the conference International Conference on Computer Vision Systems (ICVS) is only labeled Computer Vision in WikiCFP. But we know that it also belongs to Pattern Recognition, Image Processing, etc. Using the few area labels of conferences given by the CFP poster, we use Jaccard similarity between areas to find all the related labels of the conferences.

Table 2 shows the Jaccard similarity between a few research areas. Jaccard similarity between Health Informatics and E-health is 1, which indicates that these two areas are almost the same. Jaccard's similarity between Machine learning and Verification is 0.05, which indicates that these two areas are very different. Suppose L_c is the set of the labeled area(s) of a conference c such that $|L_c| > 1$. R_c is the set of related areas with Jaccard similarity greater than a threshold $th = 0.25$. We use the extended set of areas $L_c \cup R_c$ to label conference c. Table 3 shows the conferences and their extended areas after using Jaccard similarity. The related areas obtained using Jaccard similarity are highlighted in bold.

5.2 Generating Query and Access Log

We consider the keywords in titles as keyword queries and all the abstracts as the repository of sensitive documents. We consider the authors of the publications as the users who issued queries. A publication is anomalous if the conference area of that publication has very low similarity with the author's overall publication profile. To generate the IRScore and IRRank of queries, we used Apache Solr[6], which is an open-source IR system.

5.3 Finding Ground-truth Anomalous Queries

To evaluate the algorithms discussed in Section 4, we need ground-truth anomalous queries. Given the huge size of the DBLP dataset, it is difficult to manually label all the anomalous queries. In this section, we present a heuristic to generate ground truth about the queries. For this, we use the manually provided category labels, in other words, research areas of conferences to compute profiles of users and documents in terms of category labels.

Given a paper's abstract and its conference, we get the labeled areas of the conference L_c from WikiCFP and then compute the closely related areas R_c, as described in Section 5.1. We create a profile vector of the document by considering the

[6]http://lucene.apache.org/solr/

extended set of areas, where each feature of the vector is a topic area and the feature weight is one. To compute the profile vector of a user, we add up the profile vectors of all his publications, as described above. We then compute the cosine similarity between the publication and user profile vector. If the cosine similarity is below a certain threshold, we consider the particular publication as anomalous.

6 Evaluation

As introduced in Section 5, we carry out our experiments on a surrogate DBLP dataset. For this, we use the ground-truth generated in Section 5.3. The first step in detecting anomalous queries is to find Candidate Suspicious Queries (CSQ). In this regard, Figure 2 shows the relationship between the number of outliers vs SSsore. From the graph one can observe that queries with SScore less than 0.1 are not a threat as there are no outliers with SScore less than 0.1. Thus while finding CSQs, we can exclude all the queries with SScore less than 0.1. Interestingly, queries with high SScore are not a threat either. These queries seem to be issued by genuine users. This is expected for genuine users, as it is okay to access sensitive documents that are of relevance to them. Most of the outliers are concentrated around SScore value 0.2. These queries are either issued by users who are unable to form a proper query, maybe due to a lack of domain knowledge, or those who are trying to make indirect access. Since these queries are less precise, they have low SScore.

Evaluation Metric: In general outlier detection is evaluated using either the precision-recall graph or the ROC curve (Aggarwal, 2015). ROC is a plot of True Positive Rate (TPR) vs False Positive Rate (FPR). ROC has the advantage of being monotonic and more easily interpretable in terms of its lift characteristics in comparison to the precision-recall curve. ROC studies the trade-off between average-TPR and average-FPR. TPR is the number of outliers that were rightly identified while FPR measures how many non-outliers were wrongly classified as outliers. Ideally, we prefer a model that predicts all the outliers (high TPR) while being specific of not wrongly predicting normal data as outlier (low FPR). A perfect ROC curve would require the curve to stick to the left-hand side; maintaining high TPR and low FPR, and is thus said to have a high area under the curve (AUC).

Topic modeling evaluation: As discussed in Section 4.2, user representations can be computed using three topic modeling algorithms, namely TF-IDF, LDA, and TNG. From the ROC plot shown in Figure 3, we observe that TF-IDF and LDA closely follow each other. Similar to TF-IDF, LDA depends on the frequency of a word to assign a topic to it. Since abstracts are very small texts both TF-IDF and LDA have comparable outcomes here. However, TNG identifies more topic-specific terms by learning n-grams in the topics and can make a clear distinction given the small text. This is also evident from the high AUC under TNG. For the remaining evaluation, we use TNG to get the user representations.

Effect of number of neighbors: The next parameter we evaluate is the optimal number of neighbors k in computing access anomaly. The value of k depends on the nature of the data. For example in a hospital, not every department has an equal number of staff. Suppose the hospital specializes in cardiac treatment and thus has a huge cardiology department, in comparison the ophthalmology department is tiny. If we are looking for staff in the ophthalmology department, we can only get a few similar staff. If we put a big value of k, then our anomaly detection will not be accurate as we will include neighbors who are very different from the staff. The k value changes depending on the data distribution (Latourrette, 2000), it is therefore necessary to understand the data dynamics.

To find the optimal k for different data dynamics, we divided our data into three classes based on outlier density. Figures 4, 5 and 6 shows the effect of k on low, medium and high outlier density. We can observe that while $k = 2$ is an insufficient number of neighbors, $k = 15$ is too big a number to still be called 'nearest neighbors'. Both of them are bad estimators with very low AUC. Plots for all k values except $k = 6$ have close performance. For all three cases, $k = 6$ seems to give the optimal classification. For low outlier density, the number of outliers is less, therefore ROC plots are oriented towards the y-axis. However, for medium and high outlier density, the ROC curves gravitate away from the y-axis as they have a high number of outliers. All these observations are perfectly captured in Figure 7, which is the consolidated ROC plot.

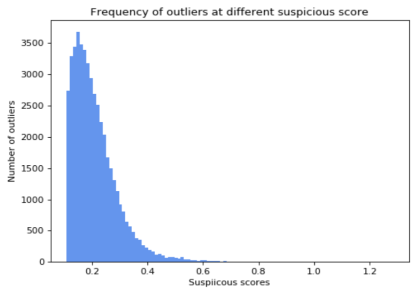


Figure 2: Outlier frequencies at varying Suspiciousness Scores

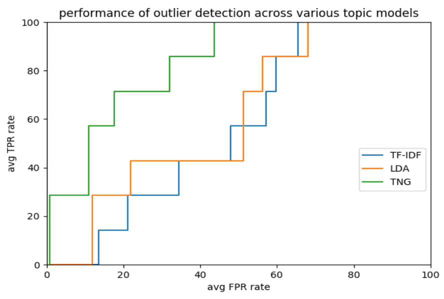


Figure 3: ROC plot for various topic models

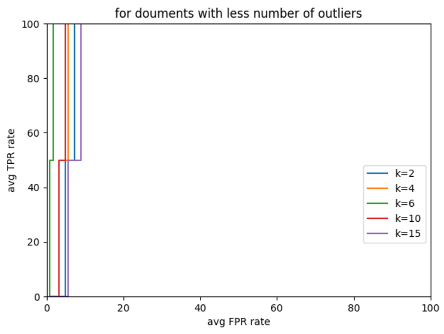


Figure 4: ROC for low outlier density

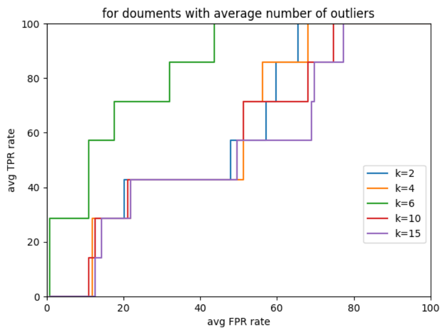


Figure 5: ROC for medium outlier density

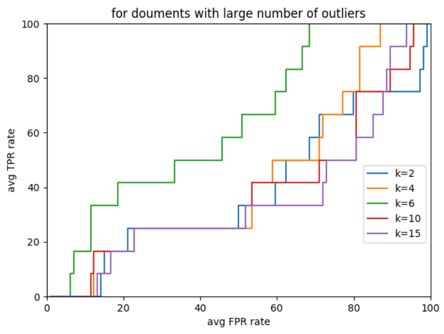


Figure 6: ROC for high outlier density

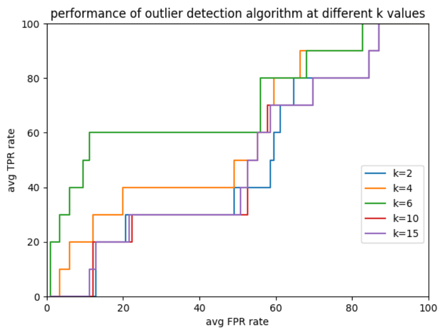


Figure 7: Overall ROC

7 Conclusion

We present one of the first of its type approaches to detect privacy violation in access of unstructured text documents using keyword queries that is mainly useful for healthcare domain. Since healthcare data is difficult to obtain, we also demonstrate the construction of a substitute dataset for healthcare. The proposed system shows promising results.

References

Charu C Aggarwal. 2015. Outlier analysis. In *Data mining*, pages 237–263. Springer.

Rakesh Agrawal, Roberto Bayardo, Christos Faloutsos, Jerry Kiernan, Ralf Rantzau, and Ramakrishnan Srikant. 2004. Auditing compliance with a hippocratic database. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 516–527. VLDB Endowment.

Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. 2002. Hippocratic databases. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 143–154. VLDB Endowment.

Elisa Bertino, Silvana Castano, and Elena Ferrari. 2001. On specifying security policies for web documents with an xml-based language. In *Proceedings of the sixth ACM symposium on Access control models and technologies*, pages 57–65. ACM.

Elisa Bertino and Elena Ferrari. 2002. Secure and selective dissemination of xml documents. *ACM*

Transactions on Information and System Security (TISSEC), 5(3):290–331.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Stefan Bottcher and Rita Steinmetz. 2006. Finding the leak: A privacy audit system for sensitive xml databases. In *Data Engineering Workshops, 2006. Proceedings. 22nd International Conference on*, pages 100–100. IEEE.

Alexander Brodsky, Csilla Farkas, and Sushil Jajodia. 2000. Secure databases: Constraints, inference channels, and monitoring disclosures. *IEEE Transactions on Knowledge and Data Engineering*, 12(6):900–919.

Ji-Won Byun, Elisa Bertino, and Ninghui Li. 2005. Purpose based access control of complex data for privacy protection. In *Proceedings of the tenth ACM symposium on Access control models and technologies*, pages 102–110. ACM.

Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, and Pierangela Samarati. 2000. Securing xml documents. In *International Conference on Extending Database Technology*, pages 121–135. Springer.

Dorothy E Denning, Peter J Denning, and Mayer D Schwartz. 1979. The tracker: A threat to statistical database security. *ACM Transactions on Database Systems (TODS)*, 4(1):76–96.

George T Duncan and Sumitra Mukherjee. 1991. Microdata disclosure limitation in statistical databases: Query sizeand random sample query control. In *null*, page 278. IEEE.

Csilla Farkas and Sushil Jajodia. 2002. The inference problem: a survey. *ACM SIGKDD Explorations Newsletter*, 4(2):6–11.

Sushil Jajodia and Catherine Meadows. 1995. Inference problems in multilevel secure database management systems. *Information Security: An integrated collection of essays*, 1:570–584.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM.

Michiharu Kudo and Satoshi Hada. 2000. Xml document security based on provisional authorization. In *Proceedings of the 7th ACM conference on Computer and communications security*, pages 87–96. ACM.

Mathieu Latourrette. 2000. Toward an explanatory similarity measure for nearest-neighbor classification. In *European Conference on Machine Learning*, pages 238–245. Springer.

Gerome Miklau and Dan Suciu. 2007. A formal analysis of information disclosure in data exchange. *Journal of Computer and System Sciences*, 73(3):507–534.

Rajeev Motwani, Shubha U Nabar, and Dilys Thomas. 2008. Auditing sql queries. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 287–296. IEEE.

Dorothy Elizabeth Robling Denning. 1982. *Cryptography and data security*. Addison-Wesley Longman Publishing Co., Inc.

Kilian Stoffel and Thomas Studer. 2005. Provable data privacy. In *International Conference on Database and Expert Systems Applications*, pages 324–332. Springer.

Xuerui Wang, Andrew McCallum, and Xing Wei. 2007. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *icdm*, pages 697–702. IEEE.

A Method to Disambiguate a Word by Using Restricted Boltzmann Machine

Nazreena Rahman[1] and **Bhogeswar Borah**[2]

[1]Assam Kaziranga University, Jorhat-785006, India
[2]Tezpur University, Sonitpur-784028, India
nazreena.rehman@gmail.com, bgb@tezu.ernet.in

Abstract

To find the correct word's sense is a great importance in many textual data related applications such as information retrieval, text mining and natural language processing. We have proposed one novel Word Sense Disambiguation (WSD) method according to its context. On the Basis of collocation extraction score, three different features are extracted for each sense definition of a target word. From the extracted features, feature vector is created. A sense matrix is formed from all the feature vectors. To enhance the sense matrix, Restricted Boltzmann Machine (RBM) is used. By using SENSEVAL and Sem Eval datasets, proposed WSD method is compared with other current systems. Practical implementation of the proposed WSD method is also shown here by applying it on query-based text summary. To implement it in query-based text summary, the method uses DUC (Document Understanding Conference) datasets. It contains newswire articles. Finally, the experimental analysis shows that our proposed WSD method outperforms many current query-based text summary systems.

1 Introduction

Disambiguation of word is much important in the fields of natural language processing and ontology. In computational linguistics, most of the languages are polysemous (Rahman and Borah, 2021b). For example, *'I am going to bank.'* The bank could be a financial institution or it could be a sloping land. In a sentence, sense of a word depends on context of the sentence (Lin et al., 2016). It is very troublesome to discover the sense of a word in computer programs because it does not possess endless information like a human being. According to Jurafsky et al. (Jurafsky, 2000), the task of selecting the correct sense of a word is known as WSD. Many text applications depend on WSD technique in their process.

The original Lesk algorithm (Lesk, 1986) uses gloss or definition of the ambiguous word (Kwon et al., 2021). Lesk algorithm can be applied only in short phrases. Wawer et al. (Wawer and Mykowiecka, 2017) developed two approaches based on the supervised and unsupervised method. The first method is an unsupervised method where log probability is computed from the sequences of word embedding vectors by considering the senses of the ambiguous word and from the context, it finds the correct sense (Rahmani et al., 2021). The second method is a supervised method where a multilayer neural network is used to find the sense of an ambiguous word. Another unsupervised method was developed for word sense disambiguation by Chaplot et al. (Chaplot and Salakhutdinov, 2018). Here, the whole text document is considered as a context for the ambiguous word. This model is based on logistic normal topic model, which adds semantic information about the synsets as its priors.

Literature survey says that different supervised, unsupervised, and knowledge-based approaches are widely used in WSD ((Navigli, 2009) (Bevilacqua et al., 2021)). A word contains different senses based on context and we need to disambiguate the target word for that given context. Word sense disambiguation method is applied in many fields like sentence similarity measure. Pawar et al. (Pawar and Mago, 2018) used 'max similarity' algorithm for WSD (Pedersen et al., 2005) where they calculated sentence similarity score. They have implemented in $Pywsd$ which is available in NLTK library in Python (Tan, 2014). However, the accuracy of this method is quite low as it does not always provide the exact sense of a word as maximum similarity does not give the surety that two senses will be exact for both sentences.

Here, an unsupervised learning algorithm named as Deep Belief Network (DBN) is applied. DBN is a probabilistic learning model having multi-layers of hidden units (Wiriyathammabhum et al., 2012). In DBN, a Restricted Boltzmann Machine (RBM) is used to train the model. Three different features are proposed to find the exact sense of a word. Finally, RBM is used to enhance the extracted features to improve the result.

2 Contribution

The main contribution is that proposed WSD method depends on three new features based on collocation extraction scores and further Restricted Boltzmann Machine is applied to enhance these three features which give better result than other existing word sense disambiguation systems. To the best of my knowledge, there is no such kind of earlier work done in disambiguation of a word by using DBN. The proposed WSD method is the original one and can be applicable in many text mining applications. This proposed WSD method can be used in semantic relatedness score calculation between two sentences as it finds the word's correct sense on the basis of context of the sentence. Further, semantic relatedness measure can be applied in many text mining applications like query-based text summarization, text clustering, plagiarism detection. The proposed WSD method is applied to query-based text summarization datasets to show its practical implementation. Query-based text summarization is different from generic summarization as it extracts essential sentences from the input text based on the user's requirement (Rahman and Borah, 2020). Therefore, to find semantic relatedness between query and input text sentence, this WSD technique is applied to get accurate relatedness score.

3 Introduction to WordNet

WordNet (Miller, 1995) is a lexical dictionary. Only content words are present in WordNet. These words are organized semantically. It is different from the traditional dictionary. Nouns, verbs, adverbs, and adjectives are present in content words. *'WordNet'* contains synonymous words set. It is known as *synset* or *synonym set*. Synonym set contains words having same meaning. For example, *shut* and *close* are synonyms. Polysemous words possess more than one *synsets*. For example, *right*; sometimes it means correct, morally good or justified and sometimes used as direction opposite to left. For each content word present in a synset, a gloss or a definition is present. Most of the content words contain more than one sense definition. In Wordnet, a word is represented as $word\#part\_of\_speech\#sense\_number$. Table 1 says about different gloss definition of word *love* along with its parts-of-speech and sense number present in WordNet.

WordNet dictionary is used here for calculating the semantic similarity or relatedness score between two words. Therefore, before finding the score, it is important to disambiguate those words.

4 Proposed Unsupervised Deep Leaning Method for Sense Detection

Process of finding the appropriate sense of a word is shown in Figure 1. Following steps are used to find the correct word's sense present in a sentence using unsupervised deep learning:

- Pre-processing: Initially, pre-processing step removes unwanted words from the text sentence. It makes the sentence a lighter one. Here, pre-processing of text document uses stop word removal. Stop words eliminates most common and unimportant words. Example of stop word are: *are*, *is*, *the* etc.

- Feature Extraction: For finding sense of a word, three features are used. It is described in section 5 and 6.

- Feature enhancement: Feature enhancement is done to improve the selection of sense for the context of the sentence. RBM is used for a feature enhancement to get the exact sense. How RBM can be applied in feature enhancement for finding word sense is described in section 7.

5 Finding Collocation Extraction Score between two Words

Collocation refers as the use or occurrence of two words together. Computational technique to find the collocation in a document or a corpus is known

Table 1: Representing content ‘love’ in WordNet

Synset(‘love.n.01’)	a strong positive emotion of regard and affection
Synset(‘love.n.02’)	any object of warm affection or devotion
Synset(‘love.n.03’)	a beloved person; used as terms of endearment
Synset(‘love.n.04’)	a deep feeling of sexual desire and attraction
synset (love.n.05’)	a score of zero in tennis or squash
Synset(‘love.v.01’)	have a great affection or liking for
Synset(‘love.v.02’)	get pleasure from
Synset(‘love.v.03’)	be enamored or in love with

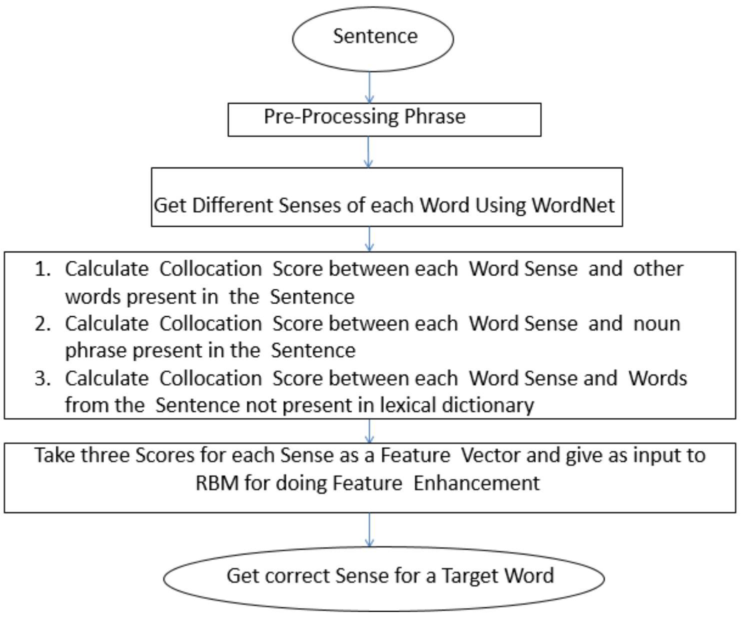

Sentence
Pre-Processing Phrase
Get Different Senses of each Word Using WordNet
1. Calculate Collocation Score between each Word Sense and other words present in the Sentence
2. Calculate Collocation Score between each Word Sense and noun phrase present in the Sentence
3. Calculate Collocation Score between each Word Sense and Words from the Sentence not present in lexical dictionary
Take three Scores for each Sense as a Feature Vector and give as input to RBM for doing Feature Enhancement
Get correct Sense for a Target Word

Figure 1: Block Diagram of Word Sense Disambiguation Method

as collocation extraction score (Rahman and Borah, 2021a). To find collocation extraction score, Wikipedia Corpus (WC) (Denoyer and Gallinari, 2006) is used. Bi-gram frequency is used to find the co-occurrence between two terms. Associativity is found in while calculating collocation. If we take the example of cat and tiger, both are semantically similar. Both are members of the feline family or superb hunters. In contrast, tiger and deer are associated as both occur frequently in language. This is known as functional relationship. Association and similarity both are not even mutually exclusive or independent. Two words tiger and deer are related two both relations to some degree (McRae et al., 2012) (Plaut, 1995). For each sense of ambiguous word, bi-gram collocation extraction score $word_1$ (McKeown and Radev, 2000) is found by calculating the frequency of words’ available in sense definition for the first word ($word_1$) with the present words in the sentence and finally maximum value is taken by the proposed WSD method.

We have taken one sentence: *Ram went to the state bank of India for depositing money*. Initially, for fining the accurate sense of word *bank*, we obtain all the senses present in WordNet. We find the collocation extraction score for each word present with the sense with the other content words present in the sentence. There are many senses present for the word *bank*. For example if I take the sense *depository financial institution*, then for each content word present in the sense *deposit, financial, institution*, we need to find the collocation extraction score with the content words present in the sentence *Ram, Go, State, India, deposit, money*. At the end, we need to take the maximum value. In this way we have to find the score for each sense. Following equation 1 is used for finding the collocation extraction score between two words. One word is selected from the gloss of *g* and other word is selected from the sentence *sen*:

$$collocation\_score(g, sen) = \frac{log(\frac{(z*SC)}{(gf*sf*span)})}{log(2)} \quad (1)$$

gf = frequency of *g* in WC
sf = frequency of *sen* in WC
z = frequency of *sen* near *g* in WC
SC= size of WC
span = width of the words (e.g. 2 to left and 2 to right of first word)

For a target word *(TaW)* exist in a sentence, the collocation extraction score (CES) of that sense is:

$$CES\,(Sense, Sentence) = max \sum_{g\in\, Sense, sen\in\, Sentence} (collocation\ score\,(g, sen)) \tag{2}$$

We find the collocation extraction score for all the senses of *TaW*. Finally, we select that sense of *TaW* for which the proposed WSD method gets the maximum collocation extraction score (Rahman and Borah, 2021b).

6 Feature Extraction for Finding Exact Sense of a Word present in a Sentence

Initially, proposed WSD method takes all the content words available in the same sentence to detect the sense of an ambiguous word. For an ambiguous word, the proposed method takes all senses present in the WordNet. For each sense, collocation extraction score is calculated for all content words. To find out the collocation extraction score, algorithm 1 shows the systematic steps:

```
Data: target word ($T_w$) and the sentence ($S_{ws}$)
Result: collocation_score of $T_w$ for each sense $s$ of $T_w$
Do the stop word removal of $S_{ws}$
Find out senses of $T_w$
for each sense ($s$) of $T_w$ do
    Do the stop word removal ($s_{swr}$) of $s$
    for each word ($w_{swr}$) of $s_{swr}$ do
        Find out the collocation extraction score between $w_{swr}$ and $S_{ws}$ by using the equation 2
    end
end
```

Algorithm 1: Collocation Extraction Score for target word's each sense with words available in the sentence

It is also seen that noun phrases always carry essential information which helps in finding the meaning of a sentence. Therefore, noun phrases are considered and find collocation extraction score. We have calculated the collocation extraction score for each sense of target word with noun phrases. Description of Algorithm 2 is given below.

```
Data: target word ($T_w$) and sentence ($S_{ws}$)
Result: collocation_score of $T_w$ for each sense $s$ of $T_w$
Do the removal of stop words $S_{ws}$
Find the noun phrases ($S_{np}$) in $S_{ws}$
Find out senses of $T_w$
for each sense ($s$) in $T_w$ do
    Do the stop word removal ($s_{swr}$) of $s$
    for each word ($w_{swr}$) of $s_{swr}$ do
        Find out the collocation extraction score between $w_{swr}$ and $S_{np}$ by using the equation 2
    end
end
```

Algorithm 2: Collocation Extraction Score of each sense of the target word with the noun phrases present in the sentence

It is also observed that sometimes, some words are not present in the WordNet, but they still can be considered as important words as they contribute to creating the context of the sentence. For example, we take three sentences: *Narendra Modi visits China. Shyam visited his uncle's house to attend the birthday party*, and *Donald Trump visited India.* The word *Visit* has different senses in WordNet. Now the proposed method will find the most suitable sense present in WordNet. In first sentence, the word *visit* is much related to *Narendra Modi.* It can understand that this *visit* must be an official visit, as Prime Minister usually goes to other foreign countries for official purpose. The same meaning is also present for the third sentence. For the second sentence, the word *visit* is related to go to see a place which is certainly not official. To find exact sense, the collocation extraction score between each sense of *visit* with *Narendra Modi*, *Shyam*, and *Donald Trump* will contribute to it. *Narendra Modi*, *Donald Trump*, and *Shyam* are not present in WordNet. That refers that, sometimes, if a word is not present in WordNet, still it helps in contributing to finding exact sense. The word *visit* is identified as a verb here.

From the above Table **??**, it is clear that the sense of *visit* word for the first and third sentence should be visit#v#4 as it matches with the context of the sentence and for the second sentence, it should be visit#v#1. The collocation extraction score between *Narendra Modi* and *office* is much higher in case of the first sentence; for the second sentence it is for *party* and *entertainment* and the third sentence it is for *Donald Trump* and *office*. This exact sense will help in finding an exact semantic relatedness score among three sentences. Therefore, the following Algorithm 3 is used to find the required collocation extraction score:

```
Data: target word ($T_w$) and the sentence
      ($S_{ws}$)
Result: collocation_score of $T_w$ for each
        sense $s$ of $S_{ws}$
Do stop word removal of $T_w$
Find the words from $S_{ws}$ not present in
  WordNet ($S_{wsnw}$)
Find out senses of $T_w$
for each sense ($s$) in $T_w$ do
    Do the stop word removal ($s_{swr}$) of $s$
    for each word ($w_{swr}$) of $s_{swr}$ do
        Find collocation extraction score
          between $w_{swr}$ and $S_{wsnw}$ with the
          help of equation 2
    end
end
```

Algorithm 3: Collocation Extraction Score of target word with words exist in the sentence but not present in WordNet

7 Use of Restricted Boltzmann Machine for Feature Enhancement to Find Correct Sense

We use Restricted Boltzmann Machine (RBM) for finding correct sense. It needs the collocation extraction score of each target word obtained from the mentioned Algorithm1, Algorithm 2, and Algorithm 3. For each target word, every sense is presented as a feature vector. For example: if a target word w has n number of senses, then it is represented as a feature vector: $w_1 = [f_{11}, f_{12}, f_{13}], w_2 = [f_{21}, f_{22}, f_{23}]............, w_n = [f_{n1}, f_{n2}, f_{n3}]$.This sense matrix will be input for the RBM. The feature vector for each target word is passed through the hidden layer where each feature vector is multiplied with respective weight and a bias value.

Initially, train the RBM with the input vector v. Now, the hidden units of RBM become deterministic. RBM has two layers: one is visible, or input layer, and the other is a hidden layer. No of units in the visible layer depends on how many senses are present for each word available in the text document. It calculates $E(h_j|v; w)$ with input v which serves again as visible units for RBM. This process can be repeated as many layers as it needs. After this unsupervised layer-wise training, back propagation is utilized to fine-tune of weights and biases (Cai et al., 2012). This unsupervised phase does not require labels. Finally, we will get a refined and enhanced matrix.

The method can be represented mathematically. Input for the RBM is a sense matrix. Each row in the matrix represents one particular sense of a word. Whole content words present in the text document are represented as a sense matrix. The sense matrix $S = (s_1, s_2, ...s_N)$ is a feature vector set contains all the three features extracted for each sense of a word s_1 (Jain and Lobiyal, 2022). The sense matrix is represented by the following Figure 2:

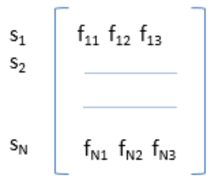


Figure 2: Sense Matrix

The input to the RBM is the set of feature vector S. It acts as a visible layer. Here, random values are selected for biases. RBM contains two hidden layers, the whole process can be represented in following equations:

$$S = (s_1, s_2, ...s_N) \tag{3}$$

Here $s_i = (f_1, f_2, f_3)$ and $i <= N$, where N is the sense number for the ambiguous word present in the text document. As the RBM has two layers, two bias values h_0 and h_1 are selected randomly.

To get a more refined matrix, RBM works in two steps. During the first phase, the new refined sense matrix is:

$$S' = (s'_1, s'_2, ...s'_N) \tag{4}$$

The above expression is obtained in the following way:

$$\sum_{i=1}^{N} s_i + h_0 \tag{5}$$

In step 2, the same procedure is followed by considering the bias h_1 and get more enhanced and refined sense matrix and which is given by:

$$S'' = (s''_1, s''_2, ...s''_N) \tag{6}$$

After obtaining refined sense matrix, a threshold value is taken for testing purpose with it. The threshold value is randomly generated for each vector. it is further tested with a For example: if the value of $f_1 >= threshold_{f1}$, then only it will be considered.

To generate optimal feature vector set, obtained feature vector sets are fine tuned by adjusting the units' weight of RBM. For optimal fine-tuning, back propagation algorithm is used. The enhanced feature vector values are added to obtain a score against each sense. Finally, the highest scored sense will be considered as the best sense for that target word. Note that the RBM will have to be freshly trained for each new content word that has to be disambiguated.

8 Experimental Analysis and Discussion

8.1 Evaluation Metric

For evaluation of the proposed word sense disambiguation method, the following equation 7 calculates the performance of the method.

$$mi = \frac{correctly\, predicted\, instances\, number}{all\, test\, instances\, number} \tag{7}$$

This mi stands for micro-average recall (Wiriyathammabhum et al., 2012).

8.2 Experiment with Word Sense Disambiguation Datasets

Proposed WSD Method is compared with other recognized and current word sense disambiguation methods where SENSEVAL-2, Sem Eval-2013 task 12 and SemEval-2015 task 13 datasets are used ((Ide and Véronis, 1998), (Wiriyathammabhum et al., 2012),(Navigli et al., 2013)). Three different features: topical local and part-of-speech- are used by Wiriyathammabhum et al. They have used different learning methods on SENSEVAL-2 dataset for word sense disambiguation (Wang et al., 2017). MFS (Most Frequent Sense) method is considered as a baseline method which chooses the major class of each word task as its prediction. Table 2 shows that proposed WSD Method performs better than existing methods. Though RBM is used by Wiriyathammabhum et al. (Wiriyathammabhum et al., 2012), but their used features are dissimilar from the proposed WSD method. Therefore, the result varies, and the proposed method performs well.

Table 2: Micro-average recall values of various learning algorithms with proposed WSD Method

Method Name	Accuracy in percentage (mi)
MFS	47.60%
1-NN	43.11%
PCA	44.45%
KPCA (polynomial)	37.50%
KPCA (Gaussian RBF)	47.71%
NB	49.95%
Logistic Regression	60.07%
MLP	59.70%
Linear SVM	60.40%
SVM (polynomial)	47.71%
SVM (Gaussian RBF)	51.02%
DBN	61.30%
Proposed WSD	72.80%

Unsupervised and supervised BabelNet-based WSD systems are used for comparison purpose (Dongsuk et al., 2018). F-score is used here as evaluation metric. BabelNet is a multilingual encyclopedic dictionary (Navigli and Ponzetto, 2012). Following widely used unsupervised systems: Moro et al. (Moro et al., 2014), Agirre et al. (Agirre et al., 2014), Apidianaki et al. (Apidianaki and Gong, 2015), Tripodi et al.(Tripodi and Pelillo, 2017), Dongsuk et al. (Dongsuk et al., 2018) and supervised systems: Zhon et al. (Zhong and Ng, 2010), Weissenborn et al. (Weissenborn et al., 2015), Raganato et al. (Raganato et al., 2017), Pasini et al. (Pasini and Navigli, 2017) are considered here. For SemEval-2013 dataset, The performance of our proposed WSD method is better than existing WSD systems. It is also seen in Table 3 that although for SemEval-2015 dataset, supervised Weissenborn et al. method performs better than our proposed WSD method but for macro-average score, proposed WSD method has shown better performance. A macro-average takes the average value of F-scores.

In comparison to other existing unsupervised (knowledge-based) methods, the proposed WSD method shows better performance. Though performance of some supervised methods are better than the existing knowledge-based methods ((Raganato et al., 2017)), but literature survey says that it is quite expensive to construct the training corpus for all the languages and words. Hence, this is one of the prominent limitation of supervised approach while applying in WSD. On other hand, WordNet ((Banerjee and Pedersen, 2003), (Chaplot et al., 2015)) is used in knowledge-based WSD system. In knowledge-based WSD systems, contextual information and semantic knowledge both are incorporated. Therefore, knowledge-based approach can disambiguate larger number of words. Conclusion can be derived from this discussion is that WSD systems which are based knowledge are more practicable and attainable than supervised WSD systems ((Chaplot et al., 2015), (Moro et al., 2014), (Chaplot and Salakhutdinov, 2018), (Dongsuk et al., 2018)). It is also tested that if anyone feature is dropped from the three proposed features, the overall performance degrades. It can be said that all three features are equally important.

9 Comparison of Results of use of Three different Features with and without the Use of Feature Enhancement

Here, we have compared the word sense disambiguation results with or without the use of feature enhancement technique. Through this comparison, it is quite clear that the feature enhancement through Restricted Boltzmann Machine helps in getting better results. Following Table 4 shows the performance of word sense disambiguation method with or without using feature enhancement in Sem Eval datasets. From the comparison 4, it is quite clear that there is a high impact of utilization of RBM in disambiguation of sense of a word.

9.1 Evaluation on Query-Based Text Summarization Datasets

9.1.1 Datasets

To further prove the performance of the proposed WSD method in practical implementations, evaluation is done on query-based text summarization datasets. WSD is widely used in text summarization. WSD helps in extracting more query oriented sentences for creating query-based text summarization. Newswire articles are taken from the Document Understanding Conference (DUC) corpora to implement WSD method. Effectiveness of the proposed method is evaluated with existing systems that perform an experimental evaluation using the Document Understanding Conference. DUC 2005 and 2006 datasets (http://duc.nist.gov) are mainly used in query-based text summarization purpose (Gervasi et al., 2019). They have complex real-life query with related text documents. Datasets contain 50 queries with 50 different topics and length of the summary is of 250 words only.

9.1.2 Evaluation of Proposed WSD Method with DUC 2005 and DUC 2006 Systems

At first, proposed WSD methods is compared with DUC 2005 and 2006 datasets. Sense of each content word is found. Table 5 and Table 6 present the different mi scores for DUC 2005 and 2006 datasets. The proposed WSD method is compared with the baseline method, along with some other existing and widely used WSD methods. Here, the baseline system represents the LESK algorithm. Table 5 and Table 6 provide the scores for different mi values. Results show that the proposed WSD method has better performane than all the existing WSD methods.

9.1.3 Evaluation of Proposed WSD Method on Query-Based Text Summarization

Now proposed WSD method is implemented for finding query-based text summary. Commonly used HSO semantic relatedness measure ((Pedersen et al., 2004), (Hirst et al., 1998)) is applied here for calculating the semantic relatedness score between query and input text sentences. Sentences are extracted based on its semantic relatedness score. The equation to find Semantic Relatedness value (S) between two words $w1$ and $w2$ is:

$$S\left((w1, s\_1, p\_1), (w2, s\_2, p\_2)\right) = 2 \times c - PL\left(w1, w2\right) - k \times DC\left(w1, w2\right) \quad (8)$$

here,
s_1= sense number of $W1$
p_1= part of speech of $W1$
s_2= sense number of $W2$
p_2= part of speech of $W2$
PL= Path Length
DC= Direction Change

Here, values of A and C are 8 and 1, respectively. The maximum value of HSO score is 16 which means two content words are same. The minimum value of HSO score is 0 which means there is no relatedness between two content words (Xia et al., 2019). For finding the Semantic Relatedness Score (S) between two sentences *s1* and *s2*, we use the

Table 3: Comparison of different BabelNet-based unsupervised and current supervised methods

Approach	System	F-score for SemEval-13	F-score for SemEval-15	Macro Avg F-score
Unsupervised (Knowledge-based)	Moro 14	66.4	70.3	68.4
	Agirre 14	62.9	63.3	63.1
	Apidianaki 15	-	64.7	-
	Tripodi 17	70.8	-	-
	Wordsim_iter$_{SRP2vSim}$ 18	75.0	65.8	70.4
	Proposed WSD	75.6	74.8	75.2
Supervised	Zhong 10	66.3	69.7	68.0
	Weissenborn 15	71.5	75.4	73.5
	Raganato 17	66.9	71.5	69.2
	Pasini 17	65.5	68.6	67.1

Table 4: Comparison results with and without the use of feature enhancement

System	F-score for SemEval-13	F-score for SemEval-15	Macro Avg F-score
Proposed WSD (without Future Enhancement)	70.6	69.3	68.6
Proposed WSD (with Future Enhancement)	75.6	74.8	75.2

Table 5: mi values for different WSD methods on DUC 2005 datasets

Method Name	Accuracy in percentage (mi)
Proposed WSD	79.2%
Original_Lesk	55%
Adapted_Lesk	60%
Cosine_Lesk	61.4%

Table 6: mi values for different WSD methods on DUC 2006 datasets

Method Name	Accuracy in percentage (mi)
Proposed WSD	81.20%
Original_Lesk	57%
Adapted_Lesk	64.34%
Cosine_Lesk	64.42%

following equation 9:

$$S(s1, s2) = \frac{\sum_{w1\in s1, w2\in s2} S((w1, s\_1, p\_1), (w2, s\_2, p\_2))}{Maximum\, relatedness\, score} \tag{9}$$

Mentioned existing WSD methods are considered again and now we use that appropriate sense for calculating S between query and input text sentences to create query-based text summary. We have taken the threshold value as 60%. It means that input sentences which are equal or greater than the threshold value are all equally important for query-based text summary creation. For comparison purpose, length of the summary is confined to 250 words for DUC datasets. Recall-Oriented Understudy for Gisting Evaluation (ROUGE) ((Lin, 2004)) is used here for evaluation purpose. ROUGE is a popular and standard intrinsic-based metric. National Institute for Standards and Technology (NIST) adapts ROUGE for summarization evaluation metric. To compare different summaries, different metrics are available in ROUGE. Quality of summary is measured in terms of overlapping units such as N-grams, word sequences, and word pairs. Different ROUGE measures ROUGE-N (N-gram co-occurrence), ROUGE-L (Longest Common Subsequence), ROUGE-W (Weighted Longest Common Subsequence), ROUGE-S (Skip-Bigram) and ROUGE-SU: (Extension of ROUGE-S) are available. ROUGE-N is a $gram_n$ recall between system-generated summary and human summary. It is based on the total number of common content words between them. Equation to find ROUGE-N is :

$$ROUGE-N = \frac{\sum_{S\in H_S}\sum_{gram_n\in S} Count_{match}(gram_n)}{\sum_{S\in H_S}\sum_{gram_n\in S} Count(gram_n)}$$

Here, N is the length of $gram_n$.

$Count_{match}(gram_n)$ says about the total common $grams_n$ co-occurring in both system and human summary and $Count(gram_n)$ gives the number of $grams_n$ present in human summary. Here, official metrics of ROUGE-1, ROUGE-2 and ROUGE-SU4 are used along with 95% confidence intervals. Tables 7 and 8 present different ROUGE scores.

Table 7: Different ROUGE values for Query-Based Text Summary on DUC 2005 datasets

Method Name	**ROUGE-1**	**ROUGE-2**	**ROUGE-SU4**
Proposed WSD	0.3812	0.0752	0.1413
Original_Lesk	0.3711	0.0624	0.1218
Adapted_Lesk	0.3768	0.0651	0.1291
Cosine_Lesk	0.3791	0.0687	0.1317

Table 8: Different ROUGE values for Query-Based Text Summary on DUC 2006 datasets

Method Name	**ROUGE-1**	**ROUGE-2**	**ROUGE-SU4**
Proposed WSD	0.4017	0.0921	0.1482
Original_Lesk	0.3992	0.0861	0.1479
Adapted_Lesk	0.4001	0.0891	0.1489
Cosine_Lesk	0.4009	0.0897	0.1494

From the evaluations, it is quite clear that the proposed WSD method helps in getting more query relevance sentences, which helps in creating a query-based text summary. Here, only the semantic relatedness measure is used. In the future, features can be increased, which will help in extracting more query related sentences.

10 Conclusion and Future Work

We have presented an unsupervised deep learning technique for detecting the word's sense. Three different features are extracted based on the collocation score. Restricted Boltzmann Machine is used to enhance the features. Proposed WSD method is implemented on word sense disambiguation datasets to compare mainly with other existing and current word sense disambiguation methods. The evaluation shows that the proposed WSD method outperforms many current methods. As this method will be used in query-based text summarization, evaluations have done on DUC datasets where the performance is much more better than many query-based text summarization methods. Experimental analysis shows better performance of our proposed WSD method than many current methods. The result attains much better due to the use of collocation based features in deep belief network. In future, the proposed WSD method can be used in many fields like question-answering, information retrieval or query-based text summarization. The proposed method will also try to work on languages other than English.

References

Eneko Agirre, Oier López de Lacalle, and Aitor Soroa. 2014. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics*, 40(1):57–84.

Marianna Apidianaki and Li Gong. 2015. Limsi: translations as source of indirect supervision for multilingual all-words sense disambiguation and entity linking. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 298–302.

Satanjeev Banerjee and Ted Pedersen. 2003. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 805–810. Acapulco, Mexico.

Michele Bevilacqua, Tommaso Pasini, Alessandro Raganato, Roberto Navigli, et al. 2021. Recent trends in word sense disambiguation: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. International Joint Conference on Artificial Intelligence, Inc.

Xianggao Cai, Su Hu, and Xiaola Lin. 2012. Feature extraction using restricted boltzmann machine for stock price prediction. In *Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on*, volume 3, pages 80–83. IEEE.

Devendra Singh Chaplot, Pushpak Bhattacharyya, and Ashwin Paranjape. 2015. Unsupervised word sense disambiguation using markov random field and dependency parser. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Devendra Singh Chaplot and Ruslan Salakhutdinov. 2018. Knowledge-based word sense disambiguation using topic models. *arXiv preprint arXiv:1801.01900*.

Ludovic Denoyer and Patrick Gallinari. 2006. The wikipedia xml corpus. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 12–19. Springer.

O Dongsuk, Sunjae Kwon, Kyungsun Kim, and Youngjoong Ko. 2018. Word sense disambiguation based on word similarity calculation using word vector representation from a knowledge-based graph. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2704–2714.

Osvaldo Gervasi, Beniamino Murgante, Sanjay Misra, Giuseppe Borruso, Carmelo M Torre, Ana Maria AC Rocha, David Taniar, Bernady O Apduhan, Elena Stankova, and Alfredo Cuzzocrea. 2019. Computational science and its applications–iccsa 2017. In *ICCSA (Conference)*, 1, pages 1–1. Springer,.

Graeme Hirst, David St-Onge, et al. 1998. Lexical chains as representations of context for the detection and correction of malapropisms. *In C. Fellbaum (Ed.), WordNet: An electronic lexical database*, chapter 13:305–332. Cambridge, Mass.: MIT Press.

Nancy Ide and Jean Véronis. 1998. Introduction to the special issue on word sense disambiguation: the state of the art. *Computational linguistics*, 24(1):2–40.

Goonjan Jain and DK Lobiyal. 2022. Word sense disambiguation using cooperative game theory and fuzzy hindi wordnet based on conceptnet. *Transactions on Asian and Low-Resource Language Information Processing*, 21(4):1–25.

Dan Jurafsky. 2000. *Speech & language processing*. Pearson Education India.

Sunjae Kwon, Dongsuk Oh, and Youngjoong Ko. 2021. Word sense disambiguation based on context selection using knowledge-based word similarity. *Information Processing & Management*, 58(4):102551.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26. ACM.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain.

Chin-Yew Lin, Nianwen Xue, Dongyan Zhao, Xuanjing Huang, and Yansong Feng. 2016. *Natural Language Understanding and Intelligent Applications: 5th CCF Conference on Natural Language Processing and Chinese Computing, NLPCC 2016, and 24th International Conference on Computer Processing of Oriental Languages, ICCPOL 2016, Kunming, China, December 2–6, 2016, Proceedings*, volume 10102. Springer.

Kathleen R McKeown and Dragomir R Radev. 2000. *Collocations. In: Dale, R., Moisl, H., Somers, H. (Eds.), Handbook of Natural Language Processing, chapter 21. Marcel Dekker, New York.* Citeseer.

Ken McRae, Saman Khalkhali, and Mary Hare. 2012. Semantic and associative relations in adolescents and young adults: Examining a tenuous dichotomy. In *In V. F. Reyna, S. B. Chapman, M. R. Dougherty, & J. Confrey (Eds.), The Adolescent Brain: Learning, Reasoning, and Decision Making*, pages 39–66. Washington, DC: American Psychological Association (APA).

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM computing surveys (CSUR)*, 41(2):1–69.

Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. Semeval-2013 task 12: Multilingual word sense disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, volume 2, pages 222–231.

Roberto Navigli and Simone Paolo Ponzetto. 2012. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Tommaso Pasini and Roberto Navigli. 2017. Train-o-matic: Large-scale supervised word sense disambiguation in multiple languages without manual training data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 78–88.

Atish Pawar and Vijay Mago. 2018. Calculating the similarity between words and sentences using a lexical database and corpus statistics. *arXiv preprint arXiv:1802.05667*.

Ted Pedersen, Satanjeev Banerjee, and Siddharth Patwardhan. 2005. Maximizing semantic relatedness to perform word sense disambiguation. Technical report, Research Report UMSI 2005/25, University of Minnesota Supercomputing Institute.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet:: Similarity: measuring the relatedness of concepts. In *Demonstration papers at HLT-NAACL 2004*, pages 38–41. Association for Computational Linguistics.

David C Plaut. 1995. Semantic and associative priming in a distributed attractor network. In *Proceedings of the 17th annual conference of the cognitive science society*, volume 17, pages 37–42. Pittsburgh, PA.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 99–110.

Nazreena Rahman and Bhogeswar Borah. 2020. Improvement of query-based text summarization using word sense disambiguation. *Complex & Intelligent Systems*, 6(1):75–85.

Nazreena Rahman and Bhogeswar Borah. 2021a. Redundancy removal method for multi-document query-based text summarization. In *2021 International Symposium on Electrical, Electronics and Information Engineering*, pages 568–574.

Nazreena Rahman and Bhogeswar Borah. 2021b. An unsupervised method for word sense disambiguation. *Journal of King Saud University-Computer and Information Sciences*.

Saeed Rahmani, Seyed Mostafa Fakhrahmad, and Mohammad Hadi Sadreddini. 2021. Co-occurrence graph-based context adaptation: a new unsupervised approach to word sense disambiguation. *Digital Scholarship in the Humanities*, 36(2):449–471.

L Tan. 2014. "pywsd: Python implementations of word sense disambiguation (wsd) technologies [software]," [online]. available: https://github.com/alvations/pywsd.

Rocco Tripodi and Marcello Pelillo. 2017. A game-theoretic approach to word sense disambiguation. *Computational Linguistics*, 43(1):31–70.

Tinghua Wang, Wei Li, Fulai Liu, and Jialin Hua. 2017. Sprinkled semantic diffusion kernel for word sense disambiguation. *Engineering applications of artificial intelligence*, 64:43–51.

Aleksander Wawer and Agnieszka Mykowiecka. 2017. Supervised and unsupervised word sense disambiguation on word embedding vectors of unambigous synonyms. In *Proceedings of the 1st Workshop on Sense, Concept and Entity Representations and their Applications*, pages 120–125.

Dirk Weissenborn, Leonhard Hennig, Feiyu Xu, and Hans Uszkoreit. 2015. Multi-objective optimization for the joint disambiguation of nouns and named entities. 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)*, volume 1, pages 596–605.

Peratham Wiriyathammabhum, Boonserm Kijsirikul, Hiroya Takamura, and Manabu Okumura. 2012. Applying deep belief networks to word sense disambiguation. *arXiv preprint arXiv:1207.0396*.

Wenlong Xia, Qingdang Meng, Qingchuan Tao, and Ray T Chen. 2019. Non-orthogonal multiple access without channel state information for similar channel conditions. *Electronics letters*, 55(8):493–495.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. *Proceedings of the ACL 2010 system demonstrations*, pages 78–83.

Encoder Decoder Approach To Automated Essay Scoring For Deeper Semantic Analysis

Priyatam Naravajhula[1], **Sreedeep Rayavarapu**[1], and **Srujana Inturi**[1]

[1]CBIT, Hyderabad-500078, India
{priyatamnaravajhula,sreedeep}@gmail.com, isrujana_cse@cbit.ac.in

Abstract

Descriptive answers have always played a major role in education of children. They are representative of student's grasp on knowledge and presentation skills. Manual evaluation of essay answers is a arduous process to human evaluators owing to limited numbers of evaluators and an out of proportional number of essays to be graded hence leading to an inefficient or an inaccurate score. It can be concluded that due to the major shift in paradigm of learning from traditional classroom education to online education engendered by COVID-19 pandemic that future assessment of education shall be online, making the solution of automatic essay scorer not only relevant, but of paramount importance. We explore several neural architecture models for the task of automated essay scoring system. Results and Experimental analysis exhibit that our model based on recurrent encoder-decoder provides for a deeper semantic analysis hence, outperforming a strong baseline in terms of quadratic weighted kappa score.

1 Introduction

The exponential advancement of deep learning in the past decade has seen its applications in a wide range of fields from molecular biology to quantum physics. This flexible nature of deep learning and neural architectures is the reason why we have seen its application to a wide array of issues in natural language processing. Automated essay scoring is one such problem which aims to find a relation between the essay written and the score assigned so that given an unseen essay, we can predict the score as accurately as possible. Essay writing forms important aspect in the academic assessment of the student, grading these essays is a laborious task therefore most of the educational organizations like Educational Testing Service (ETS) employ automated essay scorers to evaluate essays. The major pitfalls of these systems stem for the reason that they use hand crafted features to score the essay. The continuous space representations and non-linearity of neural network have provided a great potential in natural language processing. BERT and GPT-3, neural architectures developed by Google and OpenAI respectively achieve state of the art performance in NLP tasks such as word prediction, question-answering and neural machine translation.

Researchers have applied convolutional neural networks(CNNs), recurrent neural networks (RNNs), attention mechanisms (16) and a permutation of ensembles to the task of automated essay scoring. In this paper we present our encoder-decoder model that learns the relation between the essay and the score assigned by performing a deeper semantic analysis than the current existing models. By applying self-attention and non-linear layers at both encoder level and decoder level, our model is able to effectively capture the information at word level and sentence level respectively, required for scoring. We show that our model performs significantly better than our baseline neural net and finds patterns between words for a better semantic analysis.

The rest of the paper is divided into section 2 which deals with related work, section 3 which gives an idea about the task of automated essay scoring .In section 4 we present our model and all its intricacies. Section 5 gives an idea of training , section 6 deals with our experimental setup and lastly we present our results and discussion in section 7, followed by conclusion and references

2 Related Work

Some of the earliest systems of AES were dependent on handcrafted features and feature engineering. Page(1986) developed an AES tool called Project Essay Grade(PEG) by using only linguistic surface features.A well-known early example of automated essay scorer is E-Rater (Jill Burestein) (7) that employed more traditional techniques of natural language processing. The same project was released under version 2 in the year 2004 which utilizes a new set of features to represent characteristics related to organization and development, lexical complexity ,etc. All these methods shared a common regression equations for essay assessment, therefor share a common limitation of being dependent on feature engineering.

The introduction of neural networks eliminated the need for handcrafted features .Alikaniotis et al. (2016)(1) and, Taghipour and Ng, (2016)(12) presented scoring models based on LSTM. These formed some of the early examples of application of deep learning in automated essay scoring process. Particularly Taghipour and Ng, (2016)(12) presented a method to extract word level semantics by applying 1D convolution over vectors. The major limitation of the paper being usage of one-hot representations that do not extract relations as effectively as word embedding does. The usage of single layer LSTM also does not provide effective semantic relation analysis. Interestingly, Dong and Zhang,(2016)(13) presented a model involving two CNN's. In the recent years, we have seen fascinating neural architectures applied to automated essay scoring systems. Zhang and Litman,(2018)(9) proposed a novel co-attention based model that deals with source article for scoring the essay,with major limitation of not being scalable to all type of essays. Jiawei Liu et al., (2019)(14) presented a two-stage learning approach leveraging both handcrafted features and neural networks to calculate three different scores and giving a final score based on those. Siamese Neural architecture was introduced by Liang G et al,(2018)(8) where Bidirectional LSTM was used in a Siamese fashion to predict scores. In this paper, we aim to provide an end-to-end system that predicts a holistic score of the essay while ensuring that the network captures the semantic relations. Excited by the performance of encoder-decoder models in applications of NLP such as machine translation, we adopted this neural architecture for automated essay scoring system

3 Model

Our model is inspired by the neural architecture presented by Dong et al.,(2017)(3). The model presented by Dong et al; is divided into three sections:Intially, A convolution layer and attention was used to capture sentence representations. thereafter, LSTM with attention pooling for document representation was utilised. At the end, sigmoid layer was utilised for mark prediction. We have introduced a decoder layer into the network architecture, influenced by the performance of recurrent encoder-decoder layers presented by Robert Susik, (2020)(11).By doing so, our model extracts meaningful semantic relationship between sentences in the essay written by the student. Our model consists of ten layers with four layers forming the encoder architecture and the remaining six forming decoder architecture. Figure 1 depicts the architecture of our network.

3.1 Encoder Architecture

Encoder architecture consists of 4 layers:word embedding layer, convolutional layer, word level attention and an encoder LSTM layer.

3.1.1 Word Embedding Layer

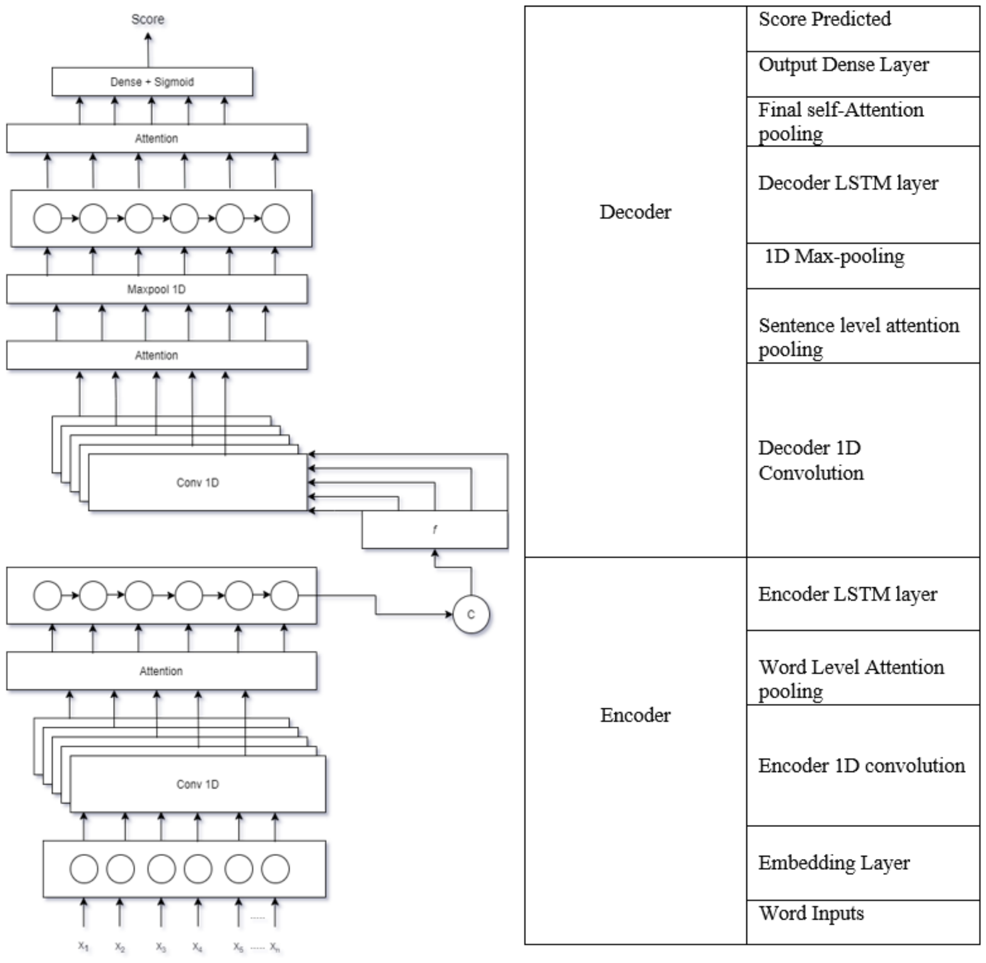
Word embeddings are used to map a word to a specific dimensional vector. We have used Glove embeddings (Pennington et al., 2014)(6) to obtain word embeddings.This particular embeddings were developed by training on six billion words from two sources It has around four hundred thousand uncased vocabulary items. The embeddings in the proposed model is restricted to fifty dimensions .The output of this layer is a matrix of dimension $L_E = R^{S \times W \times d_L}$,where S, W, dL are the number of sentences of the essay, length of the essay and embedding size .A dropout layer is applied after the embedding layer to control overfit.

3.1.2 Encoder Convolutional Layer

A 1-D convolution is performed in this layer over word representations to fetch isolated represntations in each sentence. For each word w_i in sentence, we perform a convolution:

$$k_i = a([w_i : w_{(i+l-1)}].fil_c + bs_c) \qquad (1)$$

where a is a non-linear activation function, l is the kernel size, fil_c is the filter matrix and bs_c is the bias vector. The outputs for this layer are $C_E = R^{S \times f_e \times n_C}$, where S, f_e, n_C are count of sentences in the essay, filtered lengths of sentences of the essay and number of filters used in convolutional layer .



Decoder	Score Predicted
	Output Dense Layer
	Final self-Attention pooling
	Decoder LSTM layer
	1D Max-pooling
	Sentence level attention pooling
	Decoder 1D Convolution
Encoder	Encoder LSTM layer
	Word Level Attention pooling
	Encoder 1D convolution
	Embedding Layer
	Word Inputs

Figure 1: Neural Architecture

3.1.3 Word Level Self- Attention Pooling Layer

Attention is applied following the convolutional layer , as presented in Dong et al,(2017) (3) to capture sentence representations. The attention mechanism is defined by following equations

$$mk_i = \tanh(W_m x_i + bs_m) \tag{2}$$

$$vk_i = \frac{e^{W_v mk_i}}{\sum e^{W_v mk_j}} \tag{3}$$

$$D = \sum vk_i a_i \tag{4}$$

Where W_m, w_v, bs_m are weight matrix, weight vector , bias vector respectively. mk_i, vk_i are attention weight and attention vector for a_i. The outputs for this layer are $A_E = R^{D \times n_C}$

3.1.4 Encoder Sentence Level LSTM

This layer receives the input from provious attention layer and forms the basis of first context extraction. LSTM is a modified version of recurrent units that overcome the problem of vanishing gradients effectively. (Hochreiter and Schmidhbur, 1997)(5). The power of LSTM comes from the fact that it can control the flow of information for a better sentence representation by leveraging three gates that are used to preserve or forget the information required for capturing the context of sentence representation. The output of this layer is interpreted in a manner where contextual information C is interpreted as sequence C',

$$C = \sum_{i=0}^{p_c-1} c_i \tag{5}$$

$$C' = \sum_{i=0}^{p_c/\alpha-1} \sum_{j=0}^{\alpha-1} c_{i\alpha+j} \tag{6}$$

Where $\alpha = p_c/n_x, p_c, n_x$being size of context(size of output of this layer) and size of input to this layer. The output of dimension $C = R^{S \times n_H}$, where n_H is the number of hidden states

3.2 Decoder Architecture

Decoder architecture consists of 1D convolutional layer, decoder LSTM layer, a self-attention layer and an output linear sigmoid layer.

3.2.1 Decoder Convolutional Layer

A convolutional layer is added right before the decoder to extract meaningful representations from the context C' and to also restrict the number of output channels and are is derived as follows:

$$C" = \sum_p \sum_l C'_{p,l} W_{p,l} \tag{7}$$

Where p,l are the number of kernels and length of kernel size.

3.2.2 Sentence level Attention Layer

Self-attention layer as described in section 4.1.3 is applied over 1D convolution layer. The output x of this layer is proved as input max-pool layer.

3.2.3 Decoder LSTM layer

The final context extracted from previous convolutional layer is given to this layer for capturing the semantic relations by using input output and forget gates.This layer serves as a modeling layer to construct the final sentence representation

3.2.4 Decoder Self-Attention

After obtaining the intermediate states of LSTM, a final layer of attention pooling is applied to learn the final text representation. The equations presented in 4.1.3 are also applicable here. The output O is the final text representation.

3.2.5 Linear Layer

After obtaining the final sentence representations O, a linear layer with sigmoid activation is used to predict the final output.

$$y = sigmoid(W_o O + b_o) \tag{8}$$

Where Wo and bo are weight and bias vectors.

4 Training

Automated Essay scoring is the process of evaluating the essays written by students for a particular prompt without any human intervention. Their performance is assessed by comparing the scores generated to the human-assigned gold standard scores. Rest of this section deals with the data utilized for training and the evaluation metric chosen for comparing the performance of AES systems

Table 1: ASAP Dataset Statistics

Prompt	Avg Length	Score
1	350	2–12
2	350	1–6
3	150	0–3
4	150	0–3
5	150	0–4
6	150	0–4
7	250	0–30
8	650	0–60

4.1 Data

The data that we used for our training is the one published by Hewlett Foundation for the 2012 competition titled ‘Automated Student Assessment Prize’ on Kaggle .The dataset consists of 8 prompts with three different types of essays: persuasive, source-dependent and narrative. The essays have different score ranges, being scored on average by three raters across two domains. The statistics of the dataset are given in Table 1

4.2 Evaluation Metric

The scores generated by AES systems need to be compared to ratings assigned by human-annotators. While there are many correlation metrics such as Pearson’s correlation, Spearman’s correlation, we have chosen Quadratic Weighted Kappa(QWK) score to be our evaluation metric. The main reason of this choice is because this metric is useful when it’s necessary to evaluate the possible impact of random selection in computation of standard accuracy. (Giuseppe Bonaccorso,2017)(4) IN QWK, a weighted matrix is calculated as follows

$$W(x, y) = \frac{(x-y)^2}{(U-1)^2} \tag{9}$$

Where x and y are the reference ratings and hypothesis rating respectively. U is the number of possible ratings. A matrix P is calculated where P(i,j) denotes number of essays that received a rating x from human annotators and rating y from AES. An expected count matrix K is constructed as cross vectors of two(reference and hypothesis) ratings. After normalization of K such that sum of elements of K and P are same, QWK is calculated as follows

$$\kappa = 1 - \frac{\sum_{x,y} W(x,y)P(x,y)}{\sum_{x,y} W(x,y)K(x,y)} \tag{10}$$

In our experiments, we compare QWK scores of our model to chosen baseline and performed paired t-test analysis to test the improvement obtained

4.3 Loss

MSE(Mean Square Error) calculates the average value of difference between gold standard scores y_i^* and prediction scores y_i. MSE is applied ubiquitously to regression tasks. Hence we have decided to adopt this loss function for our AES system. The following equation defines MSE, given N is the total number of samples.

$$MSE(y, y^*) = \frac{1}{N} \sum_{(i=1)^N} (y_i - y_i^*)^2 \qquad (11)$$

4.4 Optimization

In this paper, we adopted Adam optimizer (Ba et al., 2017) owing to its efficiency. Learning rate is set to 0.001, momentum to 0.9 for training our whole model. We have set Dropout rate to 0.5 to prevent overfitting.

5 Experiments

We have designed our experiments to test three hypotheses:

H1: The proposed model will perform equally or surpass baseline model on ASAP essay corpora in holistic score prediction.

H2:The proposed model will perform equally or surpass as the non-neural network baselines.

H3:Our model will have a better or at least equal semantic attention score as our baseline model.

Text preprocessing is done using NLTK , vocabulary size is restricted to 4000 consisting of most frequent words and all other words are treated as unknowns. The scores are scaled to range [0,1].(Taghipour and Ng, 2016)(12) For model training and the prediction assessment ,the predicted scores are converted back into original score ranges during model evaluation. We have divided the dataset into five folds to perform 5-fold cross validation and average QWK score across five folds on test set is reported.. In each fold,60% of data are used for training and the rest of data is equally divided between development testing.Table 2 gives a summary of hyperparameters used for training the models, taken from Dong et al.,2017(3) Best model was evaluated on development set after the completion of each epoch, this process was repeated for 100 epochs. The

Table 2: Hyperparameters

Hyperparameter	Value
Embedding dimension	50
CNN-kernel size	5
CNN-number of kernels	100
LSTM-Hidden units	100
GRU-Hidden units	100
Dropout Rate	0.5
Batch-size	100
Learning Rate	0.001
Momentum	0.9
Epochs	100

We have conducted the experiments in following software environment: Ubuntu, Python 3.7, Keras 2.4.0 using Tensorflow 2.4.1 backend. The baseline chosen for our paper is the model presented in (Dong et al, 2017). An attention based recurrent-convolutional network, where the word embedding are given to a convolutional layer to extract sentence representations. The extracted sentence representations are given as input to LSTM layer to extract semantic context. An attention layer is used for final representations. We have trained our model on this architecture and reported the QWK scores obtained. For non-neural baselines, we report the results of SVR and BLRR presented in Phandi et al, (2015)(10).They extract features such as length, prompt, and Bag of Words to classify using SVR and BLRR classifiers

6 Results and Discussion

Examining H1 hypothesis, results in Table 3 support this hypothesis. Encoder decoder model with architecture of LSTM+LSTM yields higher performance than our baseline.

The QWK scores obtained for encoder-decoder model have been shown to be significantly better on performing a paired t-test ($p < 0.05$). The reason of this high performance can be attributed to a finer text representations obtained by the architecture of encoder-decoder model and the usage of attention mechanisms at both word and sentence levels. It is interesting to note that the architecture GRU+GRU does not perform as well as its counterpart LSTM.

As we examine H2 hypothesis, QWK scores from Table 3 provide evidence to support this hypothesis, the proposed architecture outperforms or performs equally well across all non-neural archi-

Table 3: QWK scores for various architectures and baselines. The scores with statistical significant improvement (p¡0.05) are marked with "*".The highest scores for a prompt are marked in bold. Note: In system layers for decoder row, The operand before '+' is recurrent layer of encoder and the operand after is recurrent layer in decoder layer. The same notation is followed throughout the paper

ID	Architecture	System-layers	Prompts								
			1	2	3	4	5	6	7	8	Avg QWK
1	With Decoder +Attn	LSTM+LSTM	**0.808**	**0.648**	**0.686**	0.761	**0.811**	**0.823**	0.786	**0.702**	**0.753***
		GRU+GRU	0.784	0.620	0.612	**0.771**	0.757	0.791	**0.802**	0.675	0.722
2	Encoder(Baseline)+Attn	CNN +LSTM	0.796	0.644	0.593	0.752	0.761	0.782	0.762	0.699	0.719
3	Non-Neural	EASE (SVR)	0.781	0.621	0.630	0.749	0.782	0.771	0.727	0.534	0.699
		EASE (BLRR)	0.761	0.606	0.621	0.742	0.784	0.775	0.730	0.617	0.705

tectures. One contributing factor is that, the final representation in neural architectures contains more semantic information than information encoded in hand-crafted information.

Table 4: QWK score of variants of proposed model. The scores with statistical significant improvement ($p < 0.05$) are marked with '*'

I.D	System Layers	Avg.QWK
1	LSTM+GRU	**0.747***
2	GRU+LSTM	0.726
3	LSTM+BILSTM	**0.743***

Apart from the models that are reported in Table 3, we have also experimented with possible permutations of encoder-decoder architecture, as shown in Table 4.A combination of GRU and LSTM in the last layer of encoder and decoder respectively, was trained across all the prompts. Results from Table 4 show that the architecture having LSTM as final layer of encoder and GRU in the decoder(LSTM+GRU)preforms significantly better than our baseline model($p<0.05$). It is observed from Table 3 and Table 4 that final layer of encoder model is having a significant impact on the performance of the whole model. The usage of LSTM in final layer of encoder is giving significant improvement in performance than GRU. This is attributed to inefficient sentence representation of entire essay by GRU hence leading to ineffective context construction in decoder layer. We also used a bidirectional LSTM in decoder, in which the sequence of words are processed in both directions. The results of these architectures are summarized in Table 4.

Table 5 supports H3 hypothesis. In Table 5, we enlist the heatmaps of attention scores assigned by models to every word in the essay and report the average attention score. The observations are made on an essay response to prompt 5, which has been assigned a gold-standard score of 3(highest) and a predicted score of 3. The darkness of red is proportional to the attention assigned to that particular word. Prompt 5 asked the students to write about the mood created by Narciso Rodriguez in his memoir. Examining the architecture (LSTM+LSTM) closely, we can see that certain words like culinary, family, memoir are getting the highest attention while words like better, good, grateful ,love receive attention better than rest of the words. The overall average attention score of this model is higher than our baseline model, which assigns same attention to most of the words in the essay. Looking at the next architecture, (GRU+GRU) the average attention score is higher than the proposed architecture as it assigns higher attention to words better, traditions. The highest attention score is obtained by architecture (LSTM+BLSTM) that utilizes a bidirectional LSTM in the decoder layer. It assigns high attention to important words and it is also interesting to note that the model assigns high score to word collocations, some of the words like gratitude ,grateful ;culinary ,cooking received highest attention .Figure 2 depicts loss graphs for all the architectures proposed. The graphs are plotted for prompt 5, utilizing mean squared error as loss functions. The graphs show variation of loss function with 100 epochs. Figure 2 (a) shows how loss varies over 100 epochs for architecture LSTM+LSTM, while the overall trend is decreasing, intermittent pulses indicate the presence of varied samples that the model is trying to learn. The presence of LSTM in the encoder layer is giving a similar curve as observed in Figure 2 (a),(c),(e); plotted over architectures: LSTM+LSTM,LSTM+GRU,LSTM+BILSTM. Figure 2 (b),(d) plotted across architectures: GRU+GRU,GRU+LSTM, depicts a steady decrease in loss followed by a sharp convergence.

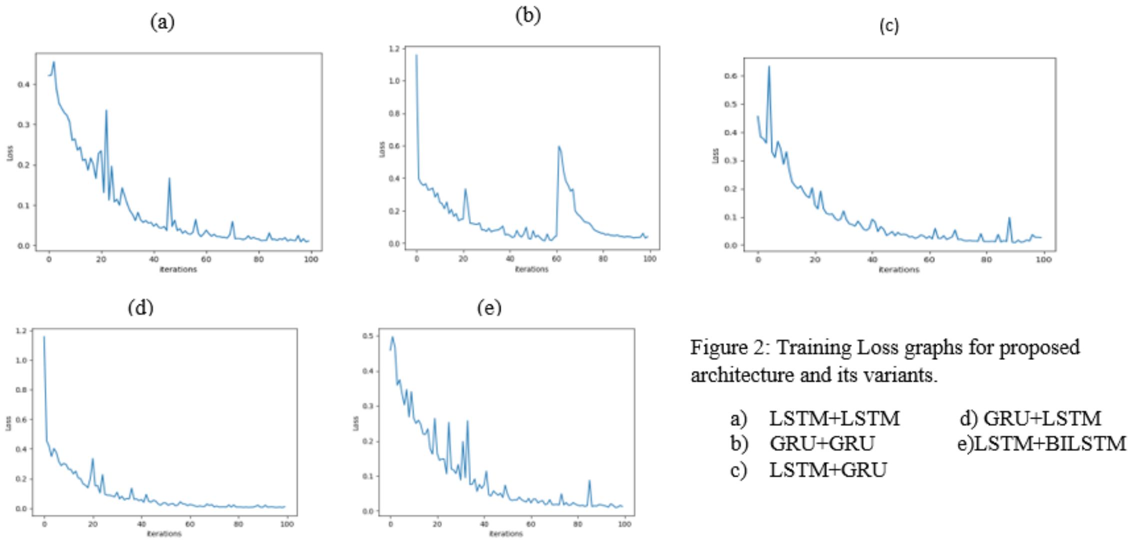


Figure 2: Training loss graphs for various architectures

Table 5: Accuracy Score and attention visualizations for prompt 5 response. The darkness of red is proportional to attention value assigned.

Architecture	System-layers	Essay	Attention
3*With Decoder	LSTM+LSTM	The mood created by the author in this memoir is gra titude Narciso Rodriguez , grateful for way his cuban parents brought him up when they had so little to be gin with @ CAPS thing that are traditions family pass ed on . One of would be there rich culinary skills and a love cooking mother father came country give bett er life even though it meant	0.492
	GRU+GRU	The mood created by the author in this memoir is gra titude Narciso Rodriguez , grateful for way his cuban parents brought him up when they had so little to be gin with @ CAPS thing that are traditions family pass ed on . One of would be there rich culinary skills and a love cooking mother father came country give bett er life even though it meant	0.501
	LSTM+BILSTM	The mood created by the author in this memoir is gra titude Narciso Rodriguez , grateful for way his cuban parents brought him up when they had so little to be gin with @ CAPS thing that are traditions family pass ed on . One of would be there rich culinary skills and a love cooking mother father came country give bett er life even though it meant	0.533
Encoder (Baseline)	CNN +LSTM	The mood created by the author in this memoir is gra titude Narciso Rodriguez , grateful for way his cuban parents brought him up when they had so little to be gin with @ CAPS thing that are traditions family pass ed on . One of would be there rich culinary skills and a love cooking mother father came country give bett er life even though it meant	0.353

6 provides a comparison between the proposed Encoder-Decoder model and the state of the art BERT model. The table provides QWK scores of BERT model taken from Rodriguez et al,**??**. The proposed model performs equally or well than BERT model in all the eight prompts.

Table 6: Comparison of QWK scores between proposed Encoder-Decoder model and BERT

Prompt	**Encoder-Decoder (LSTM+LSTM)**	**BERT**
1	0.808	0.792
2	0.648	0.679
3	0.686	0.715
4	0.761	0.801
5	0.811	0.805
6	0.823	0.805
7	0.786	0.785
8	0.702	0.595

7 Conclusion

In this paper, we have proposed a recurrent based encoder-decoder model to address the problem of automated essay scoring that outperforms the state-of -the art attention based models. The proposed model employed a decoder layer and attention mechanism to recognize germane words and sentences. Our model produces better sentence representations hence leading to a deeper semantic analysis than state of the art models. Empirical results on ASAP dataset report outperformance of our model to strong established baselines in terms of quadratic weighted Kappa score. The future scope of this to make the task of essay scoring prompt agnostic and extend beyond English language.

References

[1] Dimitrios Alikaniotis, Helen Yannakoudakis and Marek Rei. 2016. Automatic text scoring using neural networks. In Proceedings of 54th annual meeting of association for Computational Linguistics(Volume 1: Long Papers), (pp. 715-725).

[2] Diederik P. Kingma and Jimmy Ba 2017. Adam: A Method for Stochastic Optimization. Computing Research Repository. arXiv:1412.6980

[3] Fei Dong, Yue Zhang and Jie Yang. 2017. Attention baes recuurent convolutional neural networks for automatic essay scoring. 21st Conference on Computational Natural Language Learning, (pp. 153-162).

[4] Giuseppe Bonaccorso. 2017. Machine Learning Algorithms: A Reference Guide to Popular Algorithms for Data Science and Machine Learning. Packt Publishing

[5] Hochreiter Sepp and Jurgen Scmidhuber (1997). Long short-term memory. Neural computation, 1735–1780.

[6] Jeffery Pennington, Richard Socher, and Christopher D. Manning .2014. Glove:Global Vectors for word representation. Emperical Methods in Natural Language processing(EMNLP), (pp. 1532-1543)

[7] Jill Burstein, Karen Kukich, Susanne Wolff, Chi Lu, Martin Chodorow, Lisa Braden-Harder, and Mary Dee Harris.. 1998. Automated Scoring using Hybrid feature identification technique. 17th-International Conference on Computational Linguistics-Volume 1 (pp. 206-210). Association for Computational Linguistics.

[8] Guoxi Liang, Byung-Won , Dongwon Jeong , Hyun-Chul Kim and Gyu Sang Choi .2018. Automated Essay Scoring: A Siamese Bidirectional LSTM Neural Network Architecture. Symmetry.

[9] Zhang Haoran, Litman Diane 2018. Co-Attention Based Neural Network for Source-Dependent Essay Scoring. Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications (pp. 399–409). Association for Computational Linguistics.

[10] Peter Phandi, Kian Ming A Chai,abd 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, (pp. 431-439).

[11] Robert Susik. 2020. Recurrent autoencoder with sequence-aware encoding. Computing Research Repository. arXiv:2009.07349

[12] Taghipour Kaveh and Hwee Tou Ng. .2016. A Neural Approach to Automated Essay Scoring. Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing(EMNLP) (pp. 1882–1891).: Association for Computational Linguistics

[13] Dong Fei, and Yue Zhang. 2016. Automatic features for essay scoring-an empirical study. 2016 Confernece on Emprical Methods in Natural Language Processing, (pp. 1072-1077)

[14] Zhu Jiawei Liu, Yang Xu and Yaguang Zhu (2019). Automated Essay Scoring based on Two-Stage Learning. Computing Research Repository. arXiv:1901.07744

[15] Vaswani, Ashish and Shazeer, Noam and Parmar, Niki and Uszkoreit, Jakob and Jones, Llion and Gomez, Aidan N. and Kaiser, undefinedukasz and Polosukhin, Illia.(2017)Attention is All You Need

[16] Pedro Uria Rodriguez, Amir Jafari, and Christopher M Ormerod. 2019. Language models and automated essay scoring. arXiv preprint arXiv:1909.09482.

Temporal Question Generation from History Text

Harsimran Bedi
TCS Research, India

Sangameshwar Patil
TCS Research, India

Girish K. Palshikar
TCS Research, India

{bedi.harsimran, sangameshwar.patil, gk.palshikar}
@tcs.com

Abstract

Temporal analysis of history text has always held special significance to students, historians and the Social Sciences community in general. We observe from experimental data that existing deep learning (DL) models of ProphetNet and UniLM for question generation (QG) task do not perform satisfactorily when used directly for temporal QG from history text. We propose linguistically motivated templates for generating temporal questions that probe different aspects of history text and show that finetuning the DL models using the temporal questions significantly improves their performance on temporal QG task. Using automated metrics as well as human expert evaluation, we show that performance of the DL models finetuned with the template-based questions is better than finetuning done with temporal questions from SQuAD.

1 Introduction

Major events in history have always held significance for the Social Sciences community. Understanding the history of a nation, a society, an era or historic personalities involves analysing the timelines of major events that happened, their locations, the actors, and the consequences that followed. Given a set of history documents (Wikipedia pages, books, papers), it is a challenging problem to automatically extract timelines from them and to use these timelines for downstream applications such as Q&A (Bauer and Teufel, 2016; Bedi et al., 2017; Palshikar et al., 2019a,b; Gottschalk and Demidova, 2019; Hingmire et al., 2020). Another important application is to generate temporal questions from historical narrative text, which can be used for testing and improving the students' understanding of the temporal aspects of history. While much research has focused on generation of general questions from text, generation of temporal questions has received less attention (Heilman and Smith, 2010; Du et al., 2017; Pan et al., 2020; Peng et al., 2020).

We experimented with two deep learning (DL) based language models, UniLM (Dong et al., 2019) and ProphetNet (Qi et al., 2020) finetuned on SQuAD (Rajpurkar et al., 2016) for QG from history documents. The percentage of temporal questions generated and the acceptability of the questions was quite low (details in Section 4). To improve the quality and quantity of temporal questions generated, we propose linguistic knowledge based methods (*templates*). Each template analyzes the given sentence to generate a temporal question having a specific structure, by looking at the relationships among nominal and verbal events, time expressions (timex), verbs and its arguments in the dependency parse tree. Our manual evaluations show that the templates generate temporal questions with high acceptability. Then we leveraged the generated temporal questions to finetune ProphetNet and UniLM for the temporal question generation task. We evaluated the performance of the finetuned models using both domain-expert evaluation and automated metrics of BLEU-4, METEOR, and ROUGE-L. The results show that temporal questions created through our templates significantly improve the performance of DL models on the task of temporal QG.

2 Related Work

Compared to general purpose question generation (QG) and question answering (QA), the temporal aspect of QG has been relatively less explored in the literature. TEQUILA (Jia et al., 2018) is a system for temporal QA over knowledge bases (KB). It identifies temporal questions, converts them into non-temporal sub-questions and temporal constraints and then uses the underlying KB-QA engine to extract the answers to the sub-questions. Sun et al. (2018) generate and rank answers to complex questions by creating Event Graphs from text using dependency parser mainly focusing on temporal and causal relations. Both of these methods focus on temporal QA whereas our work is focused on temporal QG.

Recent trends in deep learning (DL) based methods for QG are mainly driven by neural sequence-to-sequence modeling (Qi et al., 2020; Dong et al., 2019). However, acceptability of temporal questions generated using these methods is low. Compared to other literature, the work by Peng et al. (2020) is closer to the scope of this paper. They have used triples <subject, predicate, object> from WikiData, a structured knowledge-base and a rule-based method to generate temporal questions. However, our approach uses raw input text and does not need any external KB for generating temporal questions. Further, we use the template-based questions to finetune and improve the DL methods for temporal QG.

3 Our Approach

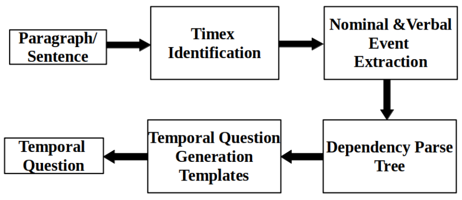


Figure 1: Temporal QG pipeline

Our method for temporal QG has two stages. In the first stage, we process the input text to extract and analyze the temporal as well as linguistic information. This information is then transformed syntactically as well as semantically using carefully designed templates to generate temporal questions. In the second stage, the generated temporal questions are used to finetune DL based models, ProphetNet and UniLM to improve their ability to generate acceptable temporal questions.

Extraction of TIMEX and Events: As shown in Figure 1, the processing pipeline of stage 1 starts with identification of tokens that indicate temporal expressions (i.e., TIMEX). We use HeidelTime (Strötgen and Gertz, 2015) to identify and normalize timex entities in the input sentence. To capture the verbal events in the input text, we make use of the past-tense propagation technique proposed by Palshikar et al. (2019a). To identify the

Algorithm 1: Algorithm for template: ``When did SUB V N?''

```
Input: S = Sentence;
PT = Parse Tree of S;
$TE_S$ = Timex Entities in S;
$NE_S$ = Nominal Events in S;
Output: Temporal question of the form:
``When did SUB V N?''

Let w be a verb in past tense in S which is not modal
  OR auxiliary verb;
if there is no such w then return;
Let V be the present tense form of w;
Let SUB be the complete text connected to w using
  DR “nsubj” in PT;
if there is no such SUB or SUB contains a pronoun
  then return;
Let P1 be a preposition in S such that P1 is connected
  to w using DR “prep” in PT AND T is a timex in
  $TE_S$ connected to P1 using DR “pobj” in PT;
if there is no such T then return;
if there is N in $NE_S$ such that N is connected to a
  preposition P2 using DR “pobj” in PT AND P2 is
  connected to w using DR “prep” in PT then
    print ``When did SUB V N?''
else if there is N in $NE_S$ such that N is connected to
  w using DR “dobj” in PT then
    print ``When did SUB V N?''
else if there is verb U in present tense in S connected
  to w using DR “xcomp” in PT AND there is N in
  $NE_S$ such that N is connected to U using DR “dobj”
  in PT then
    print ``When did SUB U N?''
```

nominal events, we make use the approach proposed by (Ramrakhiyani et al., 2021). They make use of NomBank (Meyers et al., 2004) and deverbal nouns (Gurevich et al., 2008) to identify the nominal events. Since, we specifically focus on history text in this paper, we also use a curated gazette of headwords indicating verbal and nominal events in history domain to augment the event extraction process.

Temporal Question Generation Templates: The templates are designed to probe different aspects of history text such as spatio-temporal details of an event, key players involved in it, relative temporal order among events, consequences of an event etc. We note that the set of templates is open to further extension based on the interest of historians and analysts. The proposed approach is flexible such that the questions generated by the DL based methods can be adapted to the additional templates.

Table 1 provides an overview of the templates along with examples of generated temporal questions. Due to the space constraints and ease of exposition, we focus on the template #2 `When did <Subject> <Verb> <NominalEvent>?` ; but the overall approach is similar in case of other tem-

Sr	Template	Sentence	Question
1	When did <N> happen?	During the Jassy-Kishinev Offensive of August 1944, Romania switched sides on August 23, 1944.	When did the Jassy-Kishinev Offensive happen?
2	When did <SUB> <V> <N>?	In June 1941, Hitler ordered an invasion of the Soviet Union.	When did Hitler order an invasion of the Soviet Union?
3	What happened to <SUB> after <PR> <V> <N> <P> <T>?	Gandhi launched the Quit India Movement in August 1942, after which he was arrested with other Congress lieutenants like Nehru and Patel.	What happened to Gandhi after he launched the Quit India Movement in August 1942?
4	What happened to <SUB> during <T>?	During the 1980s, Cromwell's statue was relocated outside Wythenshawe Hall, which had been occupied by Cromwell's troops.	What happened to Cromwell's statue during the 1980s?
5	Which event happened first: <N1> or <N2>?	Russia was promised Constantinople in the Constantinople Agreement of 1915. The Jews were promised a homeland in Palestine in the Balfour Declaration of 1917, but the Arabs had already been promised a sovereign state in Turkish-controlled regions.	Which event happened first: the Constantinople Agreement or the Balfour Declaration?
6	What happened to <SUB> <TM> <N> <P> <T>?	India's Prime Minister, Shastri, suffered a fatal heart attack soon after the Tashkent Agreement on January 11, 1966.	What happened to India's Prime Minister after the Tashkent Agreement on January 11, 1966?
7	When did <SUB> <VE> <O>?	By the end of 1941, German forces and the European Axis powers occupied most of Europe and North Africa.	When did the European Axis powers occupy most of Europe and North Africa?

Table 1: Overview of templates; SUB=subject, N=Nominal Event, V=Verb, P=preposition, T=Timex, N1=Nominal Event 1, N2=Nominal Event 2, TM=Temporal marker, VE=Verbal Event, O=Object

plates. We generate the dependency parse tree of the sentence using spacy (Honnibal et al., 2020) and apply the template patterns to generate temporal questions. We traverse through the part-of-speech (POS) tags and dependency relations (DR) in the parse tree of the sentence to extract phrases and tokens required to fill the relevant parameters of the templates. For instance, for template #2, we need the `Subject (SUB), Verb (V), Nominal Event (N)` with the constraint that a temporal expression (T) is appropriately associated. Algorithm 1 gives the details of how we verify the constraints and extract the parameters of the template #2. As an example, consider the sentence: `In June 1941, Hitler ordered an invasion of the Soviet Union.` Figure 2 shows its dependency parse tree and the POS-tags of tokens. From this sentence, the Algorithm 1 extracts T = `June 1941`, P = `In`, SUB = `Hitler`, V = `ordered`, N = `invasion of the Soviet Union`. Finally, after verifying the appropriate constraints, Algorithm 1 generates the question `When did Hitler order an invasion of the Soviet Union?`.

Fine-tuning DL Models for Temporal QG: The second stage of our approach overcomes the limitations of existing deep learning (DL) models for temporal QG. We use two different DL models to emphasize the flexibility and robustness of this approach. ProphetNet uses future n-gram prediction and n-stream self attention mechanism to achieve state-of-the-art performance on many NLP tasks. Unified pre-trained Language Model (UniLM) employs a shared Transformer network and specific self-attention masks. Both ProphetNet and UniLM have shown superior performance on general purpose QG task for SQuAD dataset (Qi et al., 2020; Dong et al., 2019). We observe that off-the-shelf QG models of ProphetNet and UniLM do not perform satisfactorily when used directly for temporal QG from history text (Table 4). Hence, we finetune both the DL models using the temporal questions generated by the templates in the first stage. The DL models tackle the temporal QG task as a sequence-to-sequence learning problem. The source text comprises of the input sentence and the candidate answer and the target text is the reference question. The candidate answers are generated by rule-based methods, details of which are beyond the scope of this paper. As discussed in Section 4, finetuning the models using the questions generated by our approach performs better than using questions from SQuAD dataset.

4 Experimental Evaluation

Datasets: We evaluate the proposed approach for temporal QG using history text intended for diverse audience and focusing on different topics such as historical accounts of famous personalities (e.g. Napoleon), important phenomenon (e.g., Fas-

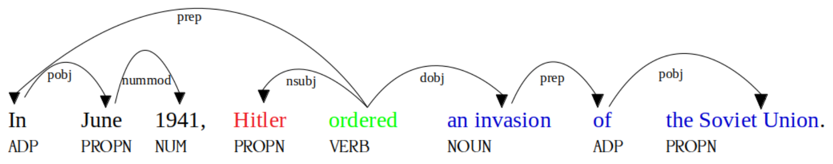


Figure 2: Dependency parse tree using Spacy for Template #2 example sentence

cism), battles, wars, and global conflicts. We use 3 chapters from history text books as well as 186 Wikipedia articles on historical topics. The pointers to the book chapters and Wikipedia articles are provided as a part of supplementary data.

Input articles	Sentences	Sentences with Time expressions	Templates	Questions	Avg. Q_{acc}
189	31538	12460	7	2480	84.07

Table 2: Template-based approach details

We use two different datasets of temporal questions for the second stage of our approach i.e. finetuning experiments with ProphetNet and UniLM. First dataset consists of 868 question generated by our template based approach for training and 217 for validation of the DL model finetuning. The second dataset consists of temporal questions (i.e., questions with explicit date-time expressions as well as implicit expressions such as *before, after, during* etc.) extracted from SQuAD dataset. The resulting subset of SQuAD dataset consists of 35794 questions as training set and 2492 questions as the validation set. The dataset used for experimentation is available for research purposes upon email request.

Finetuning Details: To evaluate the effectiveness of finetuning the DL models for temporal QG task, we use two different models, ProphetNet and UniLM.

ProphetNet: The input provided for finetuning ProphetNet is in the form of *answer [SEP] narrative* and output is the generated question. We use the hyperparameter values as suggested by the authors on their github page[1] for finetuning task. We use Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.00001. We set the dropout value as 0.1 and train for 10 epochs.

UniLM: The input provided for finetuning UniLM is in the form of *narrative [SEP] answer* and output is the generated question. Here also, we use the hyperparameter values as suggested by the authors on their github page[2] for finetuning task. We use BERT Adam optimizer (BERT version of the Adam algorithm with weight decay fix) with a learning rate of 0.00002. We train UniLM for 10 epochs.

Evaluation methodology: We employ both automated and human expert evaluation. For human evaluation, we asked experts (people well versed with English language and Global history) to mark the generated question as acceptable or non-acceptable following the human evaluation in Heilman and Smith (2010). A question is marked as acceptable if it is grammatically correct, readable, sensible and not too vague. More details can be found in the guidelines proposed by Heilman and Smith (2010). For human evaluation, we use a random sample of 100 generated questions for each experimental setting.

Q_{acc} metric is used for the percentage of generated temporal questions which are found acceptable by a human expert. $Q_{temporal}$ measures the fraction of temporal questions generated with respect to total number of generated questions. For automated evaluation, we use BLEU-4 (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005) and ROUGE-L (Lin, 2004) which are standard metrics for natural language generation tasks. The questions generated by the template based approach are used as reference questions for calculating these automated metrics.

Results and Discussion: The evaluation details of our template based approach (i.e., the stage 1 of our approach) are given in Table 2. In the corpus of 31538 sentences, there are 12460 sentences that contain a temporal expression (timex). The template based approach (i.e., the stage 1 of our approach) generates 2480 questions from this corpus with an acceptability rate of 84.07%. In this work, we have not considered the entity coreference resolution. A large number of sentences do not get considered by the templates if a slot/parameter (e.g., SUB) of a template contains a pronoun. We plan to consider entity coreference resolution (Patil et al.,

[1]https://github.com/microsoft/ProphetNet

[2]https://github.com/microsoft/unilm/tree/master/unilm-v1

Template	$\mathbf{Q_{acc}}$
template #1	94.71
template #2	82.45
template #3	78.24
template #4	73.33
template #5	60.76
template #6	89.79
template #7	80.08

Table 3: Template-wise acceptability scores (in %)

2018; Gupta et al., 2018) as part of future work.

Template wise acceptability scores are given in Table 3. Since no reference questions are available for this dataset, we evaluate the performance of template based approach through human experts. We keep aside 1000 sentences with timex that generated acceptable questions as the test set for evaluation of the second stage of finetuning DL models.

Pre-trained models of ProphetNet (denoted by P_{pre}) and UniLM (U_{pre}) for general purpose QG task are used as baselines. Let $P_{ft}^{T}, P_{ft}^{S}, P_{ft}^{TS}$ denote ProphetNet base models finetuned for temporal QG using the template based questions, SQuAD temporal questions, and the combined set of template-based as well as SQuAD temporal questions respectively. Similar notation is used for UniLM (U).

Model	B-4	M	R-L	$Q_{temporal}$	Q_{acc}
P_{pre}	0.37	0.29	0.59	53.85	80.00
P_{ft}^{T}	0.84	**0.61**	**0.93**	97.49	**99.00**
P_{ft}^{S}	0.36	0.29	0.60	70.00	91.00
P_{ft}^{TS}	0.73	0.50	0.85	98.50	**99.00**
U_{pre}	0.39	0.30	0.63	65.00	69.00
U_{ft}^{T}	**0.85**	0.60	**0.93**	**99.20**	97.00
U_{ft}^{S}	0.36	0.30	0.61	71.60	76.00
U_{ft}^{TS}	0.73	0.50	0.85	97.50	94.00

Table 4: Experimental comparison of DL models with pre-training vs. different finetuning settings (Abbr.: B-4 = BLEU-4, M = METEOR, R-L = ROUGE-L)

From Table 4, we observe that ability to generate temporal questions ($Q_{temporal}$) as well as acceptability of the generated questions (Q_{acc}) is low for both the pre-trained DL models, P_{pre} and U_{pre}. Finetuning the base models of ProphetNet as well as UniLM certainly helps to improve their performance on the temporal QG task. We note that SQuAD dataset of temporal questions is significantly larger than the set of template-based questions. Still, for both automated metrics as well as human expert evaluation, the performance of the DL models finetuned with only the template-based questions is significantly better than models that use SQuAD temporal questions.

5 Conclusion

We proposed a two-staged method for temporal QG from history text. First, we use templates motivated by linguistics and domain knowledge to carry out syntactic and semantic transformations to generate temporal questions. Then, the generated temporal questions are used to finetune DL models for QG. We experimentally validated the approach with two different DL models to demonstrate improvement due to finetuning as well as flexibility and robustness of this approach for temporal QG.

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, Ann Arbor, Michigan. Association for Computational Linguistics.

Sandro Bauer and Simone Teufel. 2016. Unsupervised timeline generation for wikipedia history articles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2343–2349.

Harsimran Bedi, Sangameshwar Patil, Swapnil Hingmire, and Girish Palshikar. 2017. Event timeline generation from history textbooks. In *Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2017)*, pages 69–77.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *arXiv preprint arXiv:1905.03197*.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. *arXiv preprint arXiv:1705.00106*.

Simon Gottschalk and Elena Demidova. 2019. Eventkg–the hub of event knowledge on the web–and biographical timeline generation. *Semantic Web*, 10(6):1039–1070.

Ajay Gupta, Devendra Verma, Sachin Pawar, Sangameshwar Patil, Swapnil Hingmire, Girish K Palshikar, and Pushpak Bhattacharyya. 2018. Identifying participant mentions and resolving their coreferences in legal court judgements. In *International Conference on Text, Speech, and Dialogue*, pages 153–162. Springer.

Olga Gurevich, Richard Crouch, Tracy Holloway King, and Valeria de Paiva. 2008. Deverbal nouns in knowledge representation. *J. Log. and Comput.*, 18(3):385–404.

Michael Heilman and Noah A Smith. 2010. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617.

Swapnil Hingmire, Nitin Ramrakhiyani, Avinash Kumar Singh, Sangameshwar Patil, Girish Palshikar, Pushpak Bhattacharyya, and Vasudeva Varma. 2020. Extracting message sequence charts from hindi narrative text. In *Proceedings of the First Joint Workshop on Narrative Understanding, Storylines, and Events*, pages 87–96.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Zhen Jia, Abdalghani Abujabal, Rishiraj Saha Roy, Jannik Strötgen, and Gerhard Weikum. 2018. Tequila: Temporal question answering over knowledge bases. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 1807–1810, New York, NY, USA. Association for Computing Machinery.

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*.

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.

A. Meyers, R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman. 2004. The nombank project: An interim report. In *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, pages 24–31, Boston, Massachusetts, USA. Association for Computational Linguistics.

Girish Palshikar, Sachin Pawar, Sangameshwar Patil, Swapnil Hingmire, Nitin Ramrakhiyani, Harsimran Bedi, Pushpak Bhattacharyya, and Vasudeva Varma. 2019a. Extraction of message sequence charts from narrative history text. In *Proceedings of the First Workshop on Narrative Understanding*, pages 28–36.

Girish Palshikar, Nitin Ramrakhiyani, Sangameshwar Patil, Sachin Pawar, Swapnil Hingmire, Vasudeva Varma, and Pushpak Bhattacharyya. 2019b. Extraction of message sequence charts from software use-case descriptions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*, pages 130–137.

Liangming Pan, Yuxi Xie, Yansong Feng, Tat-Seng Chua, and Min-Yen Kan. 2020. Semantic graphs for generating deep questions. In *ACL*.

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.

Sangameshwar Patil, Sachin Pawar, Swapnil Hingmire, Girish Palshikar, Vasudeva Varma, and Pushpak Bhattacharyya. 2018. Identification of alias links among participants in narratives. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 63–68.

Lilan Peng, Zhen Jia, Qi Dai, and Herwig Unger. 2020. A novel method of complex temporal question generation. In *Developments of Artificial Intelligence Technologies in Computation and Robotics: Proceedings of the 14th International FLINS Conference (FLINS 2020)*, pages 109–116. World Scientific.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. *arXiv preprint arXiv:2001.04063*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250.

Nitin Ramrakhiyani, Swapnil Hingmire, Sangameshwar Patil, Alok Kumar, and Girish Palshikar. 2021. Extracting events from industrial incident reports. In *Proceedings of the 4th Workshop on Challenges and Applications of Automated Extraction of Socio-political Events from Text (CASE 2021)*, pages 58–67.

Jannik Strötgen and Michael Gertz. 2015. A baseline temporal tagger for all languages. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 541–547, Lisbon, Portugal. Association for Computational Linguistics.

Yawei Sun, Gong Cheng, and Yuzhong Qu. 2018. Reading comprehension with graph-based temporal-casual reasoning. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 806–817.

CAWESumm: A Contextual and Anonymous Walk Embedding Based Extractive Summarization of Legal Bills

Deepali Jain, Malaya Dutta Borah and **Anupam Biswas**
Department of Computer Science and Engineering
National Institute of Technology Silchar, India
jaindeepali010@gmail.com,{malayaduttaborah,anupam}@cse.nits.ac.in

Abstract

Extractive summarization of lengthy legal documents requires an appropriate sentence scoring mechanism. This mechanism should capture both the local semantics of a sentence as well as the global document-level context of a sentence. The search for an appropriate sentence embedding that can enable an effective scoring mechanism has been the focus of several research works in this domain. In this work, we propose an improved sentence embedding approach that combines a Legal Bert-based local embedding of the sentence with an anonymous random walk-based entire document embedding. Such combined features help effectively capture the local and global information present in a sentence. The experimental results suggest that the proposed sentence embedding approach can be very beneficial for the appropriate representation of sentences in legal documents, improving the sentence scoring mechanism required for extractive summarization of these documents.

1 Introduction

Automatic summarization of lengthy legal documents has several benefits regarding the quick understanding of these documents for various types of users like lawyers, judges, lawmakers, and the general public (Jain et al., 2021). One of the most popular ways of performing such automatic summarization is via extractive summarization approaches, where the main idea is to extract summary-worthy sentences directly from the original documents. This process involves the appropriate representation of the individual sentences of the document, followed by the summary worthiness scoring of these sentences. The quality of sentence representation and the subsequent scoring mechanism greatly impact the overall extractive summarization performance. This motivates the need for effective sentence embedding approaches that can capture both the meaning of the individual sentences and their global context with respect to the entire document. This work proposes an improved sentence embedding approach that combines domain-specific sentence embedding with feature-based anonymous walk embeddings (AWE) of a document (Ivanov and Burnaev, 2018), which can help represent a sentence more effectively for extractive summarization.

Several research works have explored the problem of extractive summarization in the legal domain (Jain et al., 2021). CaseSummarizer (Polsley et al., 2016) is a tool which is specifically developed for summarizing legal judgment documents. In this approach, extractive summary is produced based on the frequency of words. In addition to the frequency, it also uses domain specific knowledge.

In the recent years, neural networks based approaches have shown to be very effective for extractive summarization. Most of these approaches have formulated the summarization task either as binary classification problem (Eidelman, 2019; Nallapati et al., 2017) or classification followed by ranking (Zhou et al., 2018; Narayan et al., 2018) of the sentences inside the documents. In order to perform such classification based summarization, researchers have extensively explored the problem of finding appropriate embeddings or representations of the individual sentences (Diao et al., 2020; Liu and Lapata, 2019). In this work also, our main focus is on finding the appropriate sentence representations for the summarization of legal bills.

Representing a document in terms of a sentence connectivity graph is an idea which has been very popularly applied in the general text summarization domain (Mihalcea and Tarau, 2004; Baralis et al., 2013; Li et al., 2020), because it captures the document-level global context of each sentence effectively. This suggests that if local semantics of a sentence can be combined with a graph-based global context, appropriate summarization-centric sentence embeddings can be obtained.

The key contributions of this work are given below:-

- For better sentence representation, a combination of domain-specific local embedding and graph-based anonymous random walk embedding approach is proposed.

- A detailed empirical analysis of different anonymous random walk settings are explored for finding appropriate sentence embeddings.

- A Multilayer Perceptron (MLP) based sentence summary worthiness prediction approach is presented which can make use of the improved sentence embeddings in the extractive summarization process.

Following the introduction, the organization of the rest of the paper is done as follows: A brief description of the related work is given in Section 2. A detailed description of our proposed method is given in Section 3. The evaluation strategies are presented in Section 4. The experimental results are given in Section 5 along with a detailed discussion. Finally Section 6 concludes our work, by summarizing the key findings and the potential future research directions.

2 Related Work

There are popular classical unsupervised extractive approaches in general text summarization which either utilizes the frequency-based methods (Nenkova and Vanderwende, 2005) or graph-based methods (Mihalcea and Tarau, 2004; Jing, 2000) for scoring sentences. Finally, the top scoring sentences are picked up to form an extractive summary. Several research works also find important sentences in a document, based upon the idea of Singular Value Decomposition (SVD), such as LSA (Steinberger et al., 2004). There is yet another popular classical approach in which sentences are added in a greedy manner into the summary as long as the Kullback-Lieber (KL) divergence keeps on decreasing between the document set and the summary set (Haghighi and Vanderwende, 2009). Recently a Bayesian Optimization (BO) based approach BO-Textrank has been proposed by Jain et al. (2020), in which the authors improve the Textrank algorithm for extractive summarization.

A neural network based supervised approach is proposed by Eidelman (2019), in which scores are assigned to each of the sentences of the document and the best among them are selected. The authors have formulated the sentence scoring task as a sentence classification problem for which the random ensemble and Bert models are used as classifiers to predict the important sentences for summary formation. In (Nallapati et al., 2017), authors have proposed a novel approach called SummaRuNNer, which is a Recurrent Neural Network (RNN) based approach in which the summarization task is formulated as the sequence classification task for extractive summarization of documents. Another unsupervised neural network approach is proposed by Verma and Nidhi (2017) where summary creation is done by firstly extracting the Restricted Boltzmann Machine (RBM) based features followed by a feature enhancement step.

Several random walk based approaches have been proposed in the literature for the summarization of documents represented in terms of a graph. Wang et al. (2017) have proposed an affinity-preserving random walk for the multi-document summarization problem. The summary sentences are extracted once the random walk reaches a stationary state for the purpose of summary generation. In another work, Wang et al. (2014) have proposed a random walk model in which utterances are the nodes and the relationship between two utterances is determined with the help of topic relevance, opinion relevance and structure relevance features. Finally, PageRank algorithm-based global ranking is done to select the relevant utterances to form an opinion summary. Otterbacher et al. (2005) have proposed a topic-sensitive version of Lexrank method (Erkan and Radev, 2004) where the sentence score is calculated based on the concept of random walks. The sentence score is determined by considering sentence's relevance to the query as well as it's similarity to other high scoring sentences. Apart from these works, several efficient attention mechanisms have also been proposed in the literature for handling long documents and thus achieving better performance in downstream tasks such as summarization (Zaheer et al., 2020; Beltagy et al., 2020).

From the literature review, it has been observed that most of the works ignore either the importance of domain specific knowledge or the capability to handle long documents. To deal with these shortcomings, a novel sentence representation approach is proposed in this work, which utilizes domain specific Legal-Bert embeddings of sentences along with AWE based document graph embeddings. Such combined sentence representation scheme can capture both the local sentence level information as well as the global document level information, thereby achieving better summarization of lengthy legal documents. The reason for utilizing AWE is to find the accurate vectorized representation of the entire document graph. This vectorized representation can be efficiently found using anonymous walk distribution, as proven by Micali and Zhu (2016).

3 Proposed approach

The basic steps of our proposed approach CAWE-Summ (Contextual Anonymous Walk Embedding Summarizer) to automatically generate the extractive summary of legal documents is presented in this section. Our training dataset (D^{Tr}) consists of $\{(d_1, s_1), (d_2, s_2), ..., (d_m, s_m)\}$, where (d_i, s_i) corresponds to the i^{th} document-summary pair in the dataset. The overall methodology has been depicted in Fig. 1.

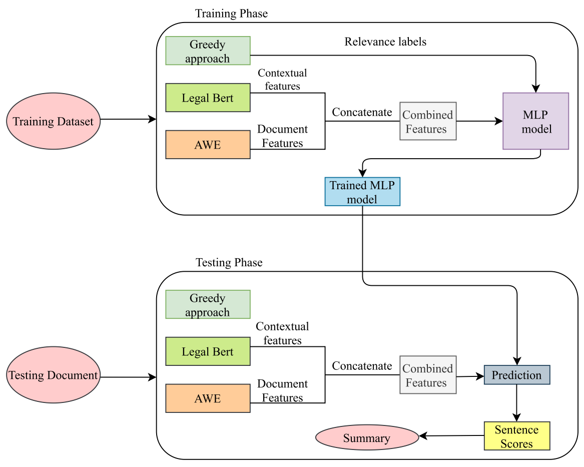


Figure 1: Overall Proposed Methodology

In the training phase, firstly, the Legal-Bert (Chalkidis et al., 2020) based sentence embeddings and Anonymous Walk Embeddings of the entire document graph are combined via concatenation operation. After this, the combined representation is used in an MLP model for learning the binary classification of summary worthiness for each of the sentences. Once the training phase is complete, we obtain a trained MLP model, which can be used at test time for predicting whether a sentence is summary-relevant or not. Each of the individual steps in Fig. 1, which depicts the proposed summarization approach, is discussed in a detailed manner in the following subsections.

3.1 Contextual Representation

The very first step is to create sentence embeddings for each of the sentences present in each document. This sentence representation is achieved through a pre-trained model. Since Bert has achieved state-of-the-art performances on several tasks (Devlin et al., 2018), researchers have started exploring the application of Bert to domain specific legal tasks as well. But the adaptation of general Bert could not perform well in legal specific tasks. Hence Chalkidis et al. (2020) has developed a legal specific Bert known as Legal-Bert. In this work, we use Legal-Bert for sentence representation which consists of 12-hidden layers, where each layer consists of 768 units. To get the contextual representation/features of sentences, the average of all the tokens in a sentences is taken. In this way, we get a vector of 768 features representing the input sentences. If the i^{th} document consists of k sentences, then it is represented as shown below:

$$\begin{aligned} d_i &= \{LB(s_1), LB(s_1), ...LB(s_k)\} \\ &= \{[s_1^1, s_1^2, ..., s_1^{768}], [s_2^1, s_2^2, ..., s_2^{768}], \\ &\quad [s_k^1, s_k^2, ..., s_k^{768}]\} \end{aligned}$$

where $LB(s_k)$ is a pretrained Legal-Bert model, applied on each sentence of a document to obtain the corresponding sentence representation.

Thus, we get the contextual representation of each sentence present in each of the documents.

3.2 Feature-based Anonymous Walk Embeddings

After obtaining the contextual representation for the input sentences, we try to enhance the representation with the help of graph representation. For this, we convert every document into Graph $G_i = (V_i, E_i)$, where V_i consists of sentences from d_i and E_i consists of direct edges between all sentences or nodes in V_i with edge weights as the similarity values between each pair of vertices. We consider cosine similarity metric for finding the similarity between each pair of vertices. The adjacency matrix representation of G_i is shown in Fig. 2.

$$\begin{bmatrix} 0 & \text{sim(LB(s}_1\text{), LB(s}_2\text{))} & \cdots\cdots & \text{sim(LB(s}_1\text{), LB(s}_k\text{))} \\ \text{sim(LB(s}_2\text{), LB(s}_1\text{))} & 0 & \cdots\cdots & \text{sim(LB(s}_2\text{), LB(s}_k\text{))} \\ \vdots & & & \vdots \\ \text{sim(LB(s}_k\text{), LB(s}_k\text{))} & \cdots\cdots & \cdots\cdots & 0 \end{bmatrix}$$

Figure 2: Adjacency matrix representation of G_i

Once the graph representation G_i is available, a p- dimensional embedding for the graph G_i can be obtained using feature-based Anonymous Walk Embeddings approach as shown below:

$AWE(G_i) = [a_1, a_2, ..., a_p]$

The main idea of AWE is to represent the random walks as a sequence of times when node in a graph was visited first, and not as a sequence of nodes (Ivanov and Burnaev, 2018). In order to understand feature-based AWE, let's first try to understand AWE. Consider the random walks as shown in Fig. 3:

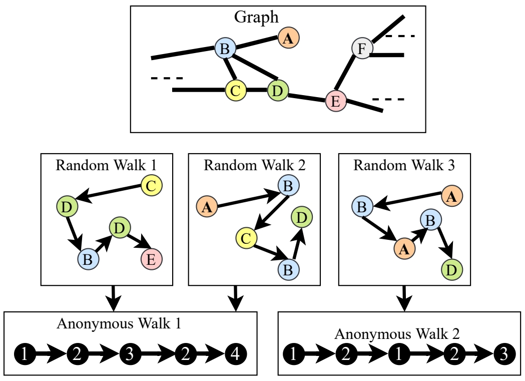


Figure 3: Illustration of anonymous walk in a unweighted directed graph

From Fig. 3, it can be observed that random walks are not represented as a sequence of nodes but as the index of a node when it appears first. These walks are known as anonymous walks because they are agnostic to the identity of nodes visited. It means that, random walks that have visited different nodes but in the same order, get the same anonymous walk representation (for example, look at random walk 1 and 2). Micali and Zhu (2016) theoretically justified that AWE, allows to encapsulate and reconstruct the structure of the entire graph irrespective of global information and therefore can be used to represent feature based embeddings for the entire network. Based on this, (Ivanov and Burnaev, 2018) came up with the feature representation of the entire network. There is a exponential growth in the number of anonymous walk with length l. For example, there are five anonymous walks w_j of length 3: $w_1 = 111$, $w_2 = 122$, $w_3 = 121$, $w_4 = 112$, $w_5 = 123$. The j^th coordinate of $AWE(G)[j]$ is the probability of anonymous walk w_j in Graph G, i.e., the probability that the anonymous walk of type j occurs in graph G. Since it is infeasible to count all the anonymous walks in a large graph, Ivanov and Burnaev (2018) have proposed an efficient sampling approach to approximate the true distribution. In this work also, we have considered the same sampling approach for finding the AWE of length 3, 4, 5 and 6, on the document graph G_i mentioned above.

3.3 Combined features

To enhance the representation of each input sentence, we propose to concatenate AWE (G_i) to each sentence embedding for the i^{th} document to obtain final d_i, thereby capturing the entire graph information as well as the local contextual information. The concatenation is done as shown below:

$$\begin{aligned} d_i = \{&[LB(s_1; AWE(G_i))], \\ &[LB(s_2; AWE(G_i))], \\ &[LB(s_k; AWE(G_i))]\} \end{aligned}$$

$$\begin{aligned} d_i = \{&[s_1^1, s_1^2, ..., s_1^{768}, a_1, a_2, ..., a_p], \\ &[s_2^1, s_2^2, ..., s_2^{768}, a_1, a_2, ..., a_p], \\ &[s_k^1, s_k^2, ..., s_k^{768}, a_1, a_2, ..., a_p]\} \end{aligned}$$

where, p is the all possible anonymous walk of length 3, 4, 5, and 6. More specifically, the possible values of p is 5, 15, 52 and 203 for lengths 3, 4, 5, and 6 respectively.

3.4 Extractive dataset building

After having all the d_i's ($i = 1, 2...., m$) for D^{Tr}, we then build an extractive training dataset for summarization using the greedy approach as proposed in (Nallapati et al., 2017). In this way, we get the dataset in the form:

$D^{TrExt} = \{[d_1, y_1], [d_2, y_2], ..., [d_m, y_m]\}$ where, each d_i is a collection of $(768 + p)$-dimensional vectors, representing each sentence of a document and y_i is a binary vector representing the relevance of each sentence in d_i for summary formation.

3.5 Summary worthiness classification task

For the purpose of training, we use an MLP (details shown in Table 1) that takes sentence embedding as an input and outputs its summary relevancy or worthiness. In order to train an MLP model, we flatten D^{TrExt} by one level to obtain $D^{TrExtMLP}$ as:

$$
\begin{aligned}
D^{TrExtMLP} = \{&([x_1, x_2, ..., x_{768+p}]^{(1)}, y^{(1)}),\\
&([x_1, x_2, ..., x_{768+p}]_{(2)}, y^{(2)}),\\
&\vdots\\
&\vdots\\
&([x_1, x_2, ..., x_{768+p}]^{(q)}, y^{(q)})\}
\end{aligned}
$$

where q is the total number of sentences across all the documents ($\approx$ 10M for BillSum dataset).

We then train the model on $D^{TrExtMLP}$, which takes $(768 + p)$-dimensional sentence embedding as input and outputs its summary worthiness. For training purpose, we consider four dense layers of nodes 768, 128, 64, 32 followed by one output layer with a sigmoid activation function. The batch size is chosen as 32, adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.001.

MLP Layers	# nodes/ Pr
FC Layer 1	768
Dropout	Pr=0.4
FC Layer 2	128
Dropout	Pr= 0.4
FC Layer 3	64
Dropout	0.4
FC Layer 4	32
Dropout	Pr=0.4
Prediction Layer(sigmoid)	1

Table 1: Number of nodes/Dropout Probabilities per layer in the MLP classification model.

3.6 Summary generation

At the time of inference, for every test document, firstly $(768 + p)$ dimensional embedding of each sentence is found, followed by the MLP based prediction of summary worthiness. The final summary formation is done by taking the top 15% sentences which is the ratio of number of words in the training documents to the number of words in the training summaries based on their summary worthiness in the order they appear in the original document.

4 Evaluation strategy

4.1 Dataset

Our proposed approach is evaluated on the BillSum dataset which is a legal specific benchmark dataset introduced by Eidelman (2019). It consists of United States (US) Congressional bills which has been divided into 18,949 training documents and 3,269 testing documents. Along with the US Congressional bills, the BillSum dataset also contains 1,237 California (CA) state bills so that the models build upon US legislatures can be tested upon new legislature as well. The training documents contain 150 sentences on an average while the training summaries contain 20 sentences on an average. The US testing dataset contains an average of 100 and 12.5 sentences in the documents and summaries respectively. The CA test dataset contains an average of 75 and 20 sentences for the CA test documents and summaries respectively.

4.2 Evaluation metric

For the purpose of evaluating the automatically generated summaries, a very popular metric known as Recall-Oriented Understudy for Gisting Evaluation (ROUGE) is considered (Lin, 2004). It counts the overlapping of n-grams between reference summaries and system generated summaries. In this work, three variants of ROUGE are considered- ROUGE-1, ROUGE-2 and ROUGE-L.

4.3 Baselines and state-of-the-art methods

We consider 8 baseline methods: Textrank (Mihalcea and Tarau, 2004), Sumbasic (Nenkova and Vanderwende, 2005), Latent Semantic Analysis (LSA) (Steinberger et al., 2004), KLSum (Haghighi and Vanderwende, 2009), Reduction (Jing, 2000), Restricted Boltzman Machines (RBM) (Verma and Nidhi, 2017), CaseSummarizer (Polsley et al., 2016) and 2 state-of-the-art methods: DOC+Sum (Eidelman, 2019) and BO-Textrank (Jain et al., 2020) (see Section 2 for brief descriptions of these methods).

Apart from these methods, we also compare our proposed approach with a Legal-Bert based summarization approach which we refer to as Contextual, in Table 2. In this approach, the Legal-Bert based local sentence embeddings are considered, in which sentence importance is predicted using the MLP model described in Section 3.5.

4.4 Experimental setup

The pretrained Legal-Bert implementation has been acquired from the Hugging-Face package [1]. Whereas, the implementation for finding AWE-based representations is publicly available from Ivanov and Burnaev (2018) and in this work the default parameter settings are utilized for the experimentation purposes. The MLP model has been implemented using the Tensorflow package (Abadi et al., 2016). Finally, the experimental results are reported in terms of the F1-Scores of the ROUGE-1, ROUGE-2 and ROUGE-L metrics, using the rouge 1.0.1 package [2].

We have run all the experiments on a Linux machine with i7 processor and RTX 2070 GPU (8GB RAM).

5 Summarization results and discussion

The ROUGE metric based summarization results for the proposed as well as the baseline and state-of-the-art approaches are depicted in Table 2. The results shows that the proposed CAWESumm approach outperforms all the baselines and state-of-the-art approaches for the extractive summarization on the BillSum test datasets. Importantly, even with the smallest length of anonymous walk embeddings, results have been improved significantly comparing to even with the legal-specific summarization baseline (CaseSummarizer) as well as with the state-of-the-art approaches. More specifically, the CAWESumm approach can obtain the best ROUGE-1, ROUGE-2 and ROUGE-L scores of 0.42827, 0.25288 and 0.41319 on the US test set respectively and 0.43120, 0.21762 and 0.36445 on the CA test set respectively.

It is important to note here that, in case of the US testing data, even though the best performances have been observed when we consider anonymous walk of length 6, still the difference between the other walk lengths are not that significant. In case of the CA testing dataset, the proposed approach has been able to outperform all the baselines and state-of-the-art methods; however here also we can see that the specific walk lengths do not have very significant impact on improving the summarization performance.

Based on the number of sentences in a document, the summarization performance of the document changes. This change in performance is depicted with the help of line charts in Fig. 4. From the line charts it can be clearly observed that when we consider small-sized documents(# of sentences ≤ 50), we can achieve much better summarization performances, across different anonymous walk lengths. On the other hand, as we get medium-sized($51 \leq$ # of sentences ≤ 100) and large-sized($100 <$ # of sentences) documents, we see that the summarization performance decrease significantly, for both the US test and CA test datasets. This is due to the fact that, always the top 15% of the high scoring sentences are picked for summary formation, and when a larger document comes as an input, the top 15% will include some low confidence predictions as well. Moreover, this causes large-sized summary formation which might be detrimental by itself.

Studying the inference time of the proposed approach can give appropriate insights into its real-time applicability. This analysis is presented in Fig. 5, with the help of a line chart diagram. From Fig. 5, it can be observed that for both the test sets, the average inference time for generating summaries is in the range of $(1 - 7)$ seconds. Another important observation is that, as the sentence embedding dimension increases, the inference time also increases sharply. This is to be expected, since larger embedding size are due to the presence of longer anonymous random walks with more number of samplings. Repeated simulation of such long walks is bound to increase the total inference time, as during inference also the AWE vectors for each sentence is needed to be calculated.

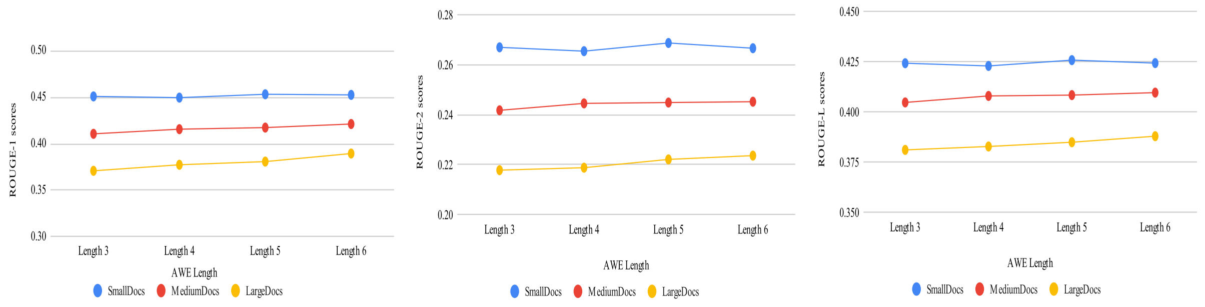
One of the key findings of this work is that the inclusion of anonymous random walk based document graph embeddings as part of the sentence embedding itself can significantly improve the overall quality of sentence representation. Such improved representation of sentences can help in the subsequent summary worthiness prediction process, as these sentences are aware of their global context in the document. The intuition behind such performance improvement is that through the learning of different anonymous walks, the embeddings are much more informative than only contextual embeddings since the anonymous walk embeddings can efficiently model the entire document. This effectiveness of AWE based graph embeddings is supported by (Micali and Zhu, 2016), where the authors prove that under sufficient samplings of anonymous random walks, entire subgraphs of the

[1] https://huggingface.co/nlpaueb/legal-bert-base-uncased

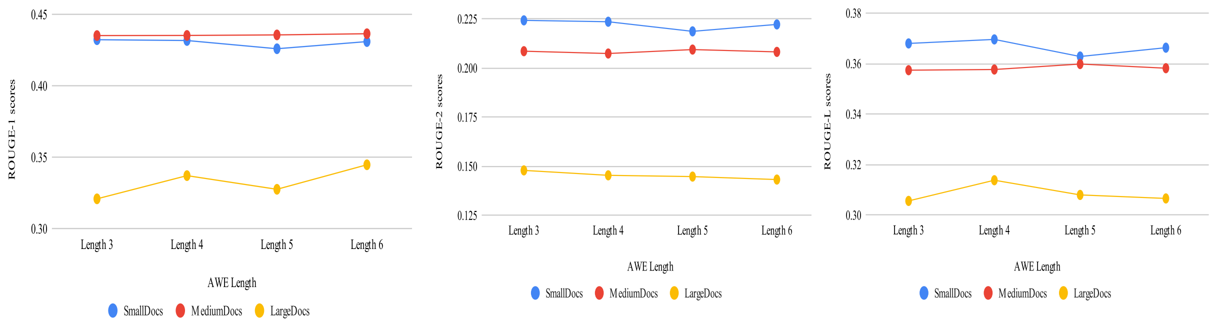
[2] https://pypi.org/project/rouge/

	Method type	Approach	US test data			CA test data		
			ROUGE-1	**ROUGE-2**	**ROUGE-L**	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**
Baseline	Unsupervised	Textrank (Mihalcea and Tarau, 2004)	0.32698	0.17939	0.33835	0.40693	0.20159	0.34574
		Sumbasic(Nenkova and Vanderwende, 2005)	0.23979	0.08106	0.22749	0.32799	0.12773	0.29769
		LSA (Steinberger et al., 2004)	0.32771	0.12888	0.28909	0.33635	0.13136	0.2971
		KLSum (Haghighi and Vanderwende, 2009)	0.26383	0.0927	0.21385	0.28002	0.10348	0.22647
		Reduction (Jing, 2000)	0.34728	0.17574	0.33046	0.39962	0.18439	0.3256
		CaseSummarizer (Polsley et al., 2016)	0.34019	0.14488	0.28507	0.36321	0.15515	0.29476
		RBM (Verma and Nidhi, 2017)	0.29710	0.10796	0.23970	0.31660	0.10074	0.24697
	Supervised	Contextual	0.38043	0.22336	0.3886	0.42100	0.20827	0.34982
		SummaRunner (Nallapati et al., 2017)	0.41604	0.22454	0.39148	0.38616	0.17467	0.32814
Proposed	Supervised	CAWESumm ($l = 3$)	0.42247	0.25002	0.41068	0.43104	**0.21762**	0.36325
		CAWESumm ($l = 4$)	0.42465	0.25058	0.41151	**0.43120**	0.21653	**0.36445**
		CAWESumm ($l = 5$)	0.42739	0.25246	0.41309	0.42730	0.21420	0.36067
		CAWESumm ($l = 6$)	**0.42827**	**0.25288**	**0.41319**	0.42998	0.21671	0.36186
State-of-the-art	Supervised	DOC + SUM (Eidelman, 2019)	0.4080	0.2383	0.3373	0.3965	0.2114	0.3405
	Unsupervised	BO-Textrank (Jain et al., 2020)	0.356	0.172	0.312	0.404	0.194	0.327

Table 2: Comparison of proposed CAWESumm approach for different AWE lengths (l), with baseline and state-of-the-art approaches.



(a) US Test



(b) CA Test

Figure 4: Document length wise ROUGE scores

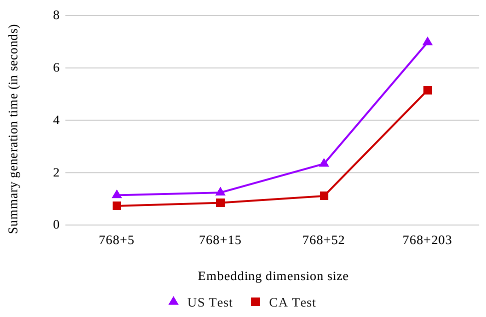


Figure 5: Average running time to generate summaries for BillSum testing dataset where 5,15,52 & 203 are all possible anonymous walks for l=3, 4, 5 & 6 respectively.

underlying graph can be reconstructed with the help of such walks in that region. In our case, this result ensures that the AWE obtained on the document graph can effectively represent the original input document. Moreover, since the MLP model has been trained to recognize summary worthy embeddings during the training process, the interaction between the contextual and document graph embedding features can be effectively modeled for summarization.

6 Conclusion

Due to the lengthy nature of legal bills, it becomes very difficult to capture the important information of the documents. To overcome this difficulty, extraction of summary worthy sentences for automatic summarization has been explored in the literature. However, in order to efficiently identify summary worthy sentences, they need to be appropriately represented in the form of numerical vectors. In this work, we propose to capture a sentence's local as well as global context information in the form of embeddings via contextual and anonymous walk embeddings. From the experimental results, it has been found that, when we incorporate the global document level information with the sentence's local information, a significant improvement can be obtained in terms of ROUGE scores. The experimental results suggests that the anonymous walk embeddings are very effective in capturing the entire document graph information, and can enhance the representation of a sentence for its summary worthiness prediction. Such improved sentence representation are able to significantly improve the extractive summarization of the document.

To further improve the representation learning, leveraging embeddings in the form of hierarchical structure of the entire legal documents will be part of our future work. Moreover, a more end-to-end graph based approach can also be studied, by considering the emerging area of graph neural networks.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283.

Elena Baralis, Luca Cagliero, Naeem Mahoto, and Alessandro Fiori. 2013. Graphsum: Discovering correlations among multiple terms for graph-based summarization. *Information Sciences*, 249:96–109.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. Legal-bert: The muppets straight out of law school. *arXiv preprint arXiv:2010.02559*.

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*.

Yufeng Diao, Hongfei Lin, Liang Yang, Xiaochao Fan, Yonghe Chu, Di Wu, Dongyu Zhang, and Kan Xu. 2020. Crhasum: extractive text summarization with contextualized-representation hierarchical-attention summarization network. *Neural Computing and Applications*, 32(15):11491–11503.

Vladimir Eidelman. 2019. Billsum: A corpus for automatic summarization of us legislation. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 48–56.

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.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of human language technologies: The 2009 annual conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370.

Sergey Ivanov and Evgeny Burnaev. 2018. Anonymous walk embeddings. In *International conference on machine learning*, pages 2186–2195. PMLR.

Deepali Jain, Malaya Dutta Borah, and Anupam Biswas. 2020. Fine-tuning textrank for legal document summarization: A bayesian optimization based approach. In *Forum for Information Retrieval Evaluation*, pages 41–48.

Deepali Jain, Malaya Dutta Borah, and Anupam Biswas. 2021. Summarization of legal documents: Where are we now and the way forward. *Computer Science Review*, 40:100388.

Hongyan Jing. 2000. Sentence reduction for automatic text summarization. In *Sixth Applied Natural Language Processing Conference*, pages 310–315.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Wei Li, Xinyan Xiao, Jiachen Liu, Hua Wu, Haifeng Wang, and Junping Du. 2020. Leveraging graph to improve abstractive multi-document summarization. *arXiv preprint arXiv:2005.10043*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries acl. In *Proceedings of Workshop on Text Summarization Branches Out Post Conference Workshop of ACL*, pages 2017–05.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*.

Silvio Micali and Zeyuan Allen Zhu. 2016. Reconstructing markov processes from independent and anonymous experiments. *Discrete Applied Mathematics*, 200:108–122.

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.

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*.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. *arXiv preprint arXiv:1802.08636*.

Ani Nenkova and Lucy Vanderwende. 2005. The impact of frequency on summarization. *Microsoft Research, Redmond, Washington, Tech. Rep. MSR-TR-2005*, 101.

Jahna Otterbacher, Gunes Erkan, and Dragomir Radev. 2005. Using random walks for question-focused sentence retrieval. In *Proceedings of human language technology conference and conference on empirical methods in natural language processing*, pages 915–922.

Seth Polsley, Pooja Jhunjhunwala, and Ruihong Huang. 2016. Casesummarizer: a system for automated summarization of legal texts. In *Proceedings of COLING 2016, the 26th international conference on Computational Linguistics: System Demonstrations*, pages 258–262.

Josef Steinberger, Karel Jezek, et al. 2004. Using latent semantic analysis in text summarization and summary evaluation. *Proc. ISIM*, 4:93–100.

Sukriti Verma and Vagisha Nidhi. 2017. Extractive summarization using deep learning. *arXiv preprint arXiv:1708.04439*.

Kexiang Wang, Tianyu Liu, Zhifang Sui, and Baobao Chang. 2017. Affinity-preserving random walk for multi-document summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 210–220.

Zhongqing Wang, Liyuan Lin, Shoushan Li, and Guodong Zhou. 2014. Random walks for opinion summarization on conversations. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 430–437. Springer.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. In *NeurIPS*.

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. *arXiv preprint arXiv:1807.02305*.

Multi-document Text Summarization using Semantic Word and Sentence Similarity: A Combined Approach

Rajendra Kumar Roul
Thapar Institute of Engineering and Technology
Patiala, Punjab, India
raj.roul@thapar.edu

Abstract

The exponential growth in the number of text documents produced daily on the web poses several difficulties to people who are responsible for collecting, organizing, and searching different textual content related to a particular topic. Automatic Text Summarization is effective in this direction, as it can evaluate a large number of documents and extract essential information. However, the limits of automatic text summarization must be overcome by devising practical solutions. Even though current research efforts focus on this direction for future advances, they still face numerous obstacles. This work suggests a combined semantic-based word and sentence similarity technique to summarise a corpus of text documents. KL-divergence approach is used to organize the sentences in the final summary. Experimental work is conducted using DUC datasets, and the obtained results are promising.

1 Introduction

With the widespread adaptation of technology, a large number of documents are getting digitized, resulting in a rapid influx of textual data. This data often contains crucial information; however, absorbing all this information can be difficult and time-consuming. Automatic Text Summarization (ATS) is the process of condensing data into useful and comprehensible information. By distilling out meaningful details, ATS makes referring documents much more efficient. ATS can be done in two ways: *Extractive* and *Abstractive*. Extractive summarization selects sentences of importance directly from the source text, which can either be within a single document (called single document text summarization) or a group of documents (called multi-document text summarization)(Gupta and Lehal, 2010)(Roul and Arora, 2019). On the other hand, abstractive text summarization is an understanding of the main concept of its expression in clear natural language. When abstraction is used for text summarization in deep learning issues, it can overcome the extractive method's grammatical inconsistencies.

1.1 Motivation

There has already been a vast amount of research on text summarization such as 'graph-based summarization (Elbarougy et al., 2020)', 'clustering-based summarization(Wang et al., 2011)(Roul et al., 2016)', 'machine learning based summarization(Roul et al., 2017)(Abdi et al., 2018)', 'summarization based on Fuzzy logic' (Suanmali et al., 2009), topic-modeling based summarization(Roul et al., 2019)(Alami et al., 2021)(Roul, 2021) etc. As listed below, all of these existing text summarising approaches have some common limitations:

i. Two different sentences made up of completely different words can share a similar meaning, and it should be taken care of when the summary is generated.

ii. Stop-words like 'a,' 'an,' 'the,' 'of,' and so on are often excluded from surface matching algorithms since they are relatively prevalent throughout all articles in the collection. However, these words play a significant part in calculating sentence similarity since they provide structural information that is used to infer the content of the phrase, and hence they should not be ignored.

iii. The significance of the words in the scope of the sentence is ignored.

iv. When computing the similarity of sentences, giving equal weight to each word is still lacking.

This study considers the extractive approach towards achieving summarization and presents a snapshot of the content of a group of related documents. Semantic-based word and sentence similarities are combined to generate a coherent summary at the end.

1.2 Contribution

The following is a summary of the paper's contributions:

i. The problem of organizing and logically displaying the gathered data has not yet received attention. The suggested method computes each sentence's cohesiveness score to eliminate redundancy and picks the top 'm' percent of sentences based on the cohesion score to generate the coherent summary.

ii. Every word of the generated coherent summary gets equal algebraic treatment by considering modified harmonic mean.

iii. The suggested method, which includes all the stop-words, treats each word in a sentence separately according to its semantic structure.

iv. The proposed approach computes the semantic similarity between the sentences to get a more information-rich coherent summary.

v. In the generated summary, all the sentences are arranged as per their importance using the Kullback-Leibler divergence technique.

Empirical results show that the suggested approach is more efficient than the existing extractive text summarization approaches.

2 Proposed Approach

Assume a corpus P consists of D number of documents. At first, all D documents are merged into a single huge collection. All the sentences of this huge collection are extracted to form a set of sentences $S = \{s_1, s_2, s_3, \cdots, s_n\}$. Below steps discuss how the coherent summary is generated from the corpus P.

1. Word similarity calculation:
Semantic similarity between two words (a and b) is calculated using WordNet (Miller, 1995) having 206942 words and 117660 synsets. Figure 1 shows the hierarchy of these synsets (or concepts) of the semantic database of the WordNet. The symbol '$\cdots$' is used to represent more synonym words of a synset. To extract the information from the semantic database, WordNet.Net[1], a public framework is used here. The semantic sim-

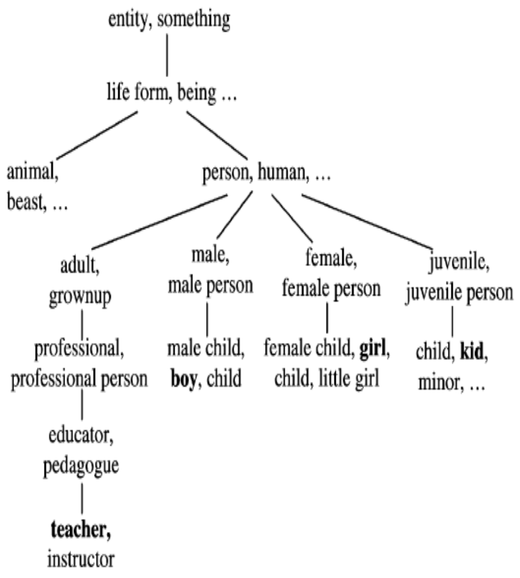


Figure 1: Hierarchical Semantic Net

ilarity $sim(a, b)$ is calculated using two functions:

- minimum path length (min_path)
- depth of the subsumer ($depth\text{-}sub$)

$$sim(a, b) = function(f_1(min\_path), f_2(depth\text{-}sub)) \quad (1)$$

i. Computing minimum path length:
There are 3 posibilities as mentioned below while computing the $sim(a, b)$:
   - a and b are belong to the same synset: since they have the same meaning, a semantic path length of zero is allocated between them.
   - a and b have not belonged to the same synset: here the shortest path between the two synsets is calculated by 'max-similarity' algorithm (Pedersen et al., 2005) using Pywsd[2].
   - a and b do not belong to the same synset, but their corresponding

[1]http://en.wikipedia.org/wiki/Brown_Corpus
[2]https://github.com/alvations/pywsd

synset consists of one or more common words: in this circumstance, a semantic path length of one is allocated since both synsets share some of the same terms.

In light of the three situations presented above, the $f_1(min\_path)$ of equation 1 is fix to be a steadily reducing function as shown in equation 2.

$$f_1(min\_path) = e^{-\alpha(min\_path)} \tag{2}$$

here $\alpha \in [0, 1]$ is constant.

ii. Depth of the subsumer computation:
The depth of the subsumer is determined by counting the levels from the subsumer to the hierarchical net's top. Words in the top layers of the hierarchy have a broader meaning and fewer semantic concepts than words in the lower layers. When computing the similarity, this behavior must be taken into account. Thus, it is necessary to scale up the $sim(a, b)$ for subsuming words at bottom layers, and for subsuming words at higher layers, one needs to scale down the $sim(a, b)$. This shows $f_2(depth\text{-}sub)$ of equation 1 should be monotonically increasing function as shown illustrated in equation 3.

$$f_2(depth\text{-}sub) = \frac{e^{\beta.depth\text{-}sub} - e^{-\beta.depth\text{-}sub}}{e^{-\beta.depth\text{-}sub} + e^{\beta.depth\text{-}sub}} \tag{3}$$

where $\beta \in [0, 1]$ is a smoothing factor, and it determines the contribution of depth of subsumer. With respect to α of equation 2,The percentage contribution of subsumer depth reduces as *beta* rises. The word's depth in the hierarchy is not considered when $\beta > \infty$ (Shepard, 1987). The optimum values of β and α are set to 0.46 and 0.2 respectively (Erkan and Radev, 2004).

iii. Finally, semantic similarity between a and b is measured using equation 4.
$sim(a, b) =$

$$e^{-\alpha.min\_path} * \frac{e^{\beta.depth\text{-}sub} - e^{-\beta.depth\text{-}sub}}{e^{\beta.depth\text{-}sub} + e^{-\beta.depth\_sub}} \tag{4}$$

The value of $sim(a, b) \in [0, 1]$.

2. *Word score calculation based on modified harmonic mean*:
The harmonic mean can't be determined without taking into account all of the words in the corpus. It gives each word equal weight and is excellent for qualitative data. The modified Harmonic Mean (HM) formula is used to produce a ranking score for each word in relation to the total corpus, as shown in equation 5.

$$HM_q = \frac{n-1}{\sum\limits_{p,p\neq q} \frac{1}{sim(a,b)+k}} \tag{5}$$

n indicates the number of words in P, $sim(a, b)$ represents the similarity score between the two words a and b as shown in equation 4. Except for the reflexive pair, all of the pairs of words are summed up. k is a factor that must be included in every similarity score in the algorithm to ensure that the score, when divided by 1, does not provide an exception.

3. *Selection of representative words*:
Upon calculating the modified harmonic of every word in P, the top $l\%$ words[3] are saved in a list L_{rep} as representative words of the corpus P. Now, as stated in equation 6, the cosine-similarity ($cos\text{-}sim$) between s and the list L_{rep} is calculated.

$$cos\text{-}sim(s, L_{rep}) = \frac{s.L_{rep}}{||s|| * ||L_{rep}||} \tag{6}$$

Sentences having cosine similarity more than 0.75[4] are considered.

4. *Calculation of sentence similarity*:
The following steps are used to measure similarity between two sentences :

i. A combined word set construction:
To compare the similarity of two sentences s_1 and s_2, create a combined set of words $J_s = \{w_1, w_2, \cdots, w_n\}$, where each w_i is an unique word from s_1 and s_2. This means there are no common terms in J_s between s_1 and s_2. Because they carry syntactic information, J_s also contains function words. The word form is maintained in the same way as it appears in the sentence.

[3] chosen by the experiment
[4]decided by experiment

ii. Similarity between sentences:
The structural semantic vector ($lsv_i, i \in [1,2]$ of s_1 and s_2 is computed first. Each lsv_i entry corresponds to a word in J_s. For every $w \in J_s$, the following steps are utilised to calculate the $lsv_i, i = 1$ for s_1 (denoted as lsv_1). Prior to the commencement of the procedure, a semantic vector sv is considered, with every entries set to zero.

case-a'. $w \in s_1$: the sv entry corresponding to s_1 is set to 1. This value is indicated in equation 7.

$$lsv_1 = I(w)^2 * sv \tag{7}$$

case-b'. $w \notin s_1$: an identical word (designated as $\overline{w}$) is found in $s1$ by analyzing the semantic relatedness of w to each word in s_1 (semantic relatedness is determined using equation 4). The related entry in the (sv) is fix to the estimated similarity, if it exceeds a per-defined threshold value[5], otherwise it is set to zero. Equation 8 shows the detail.

$$lsv_1 = I(w) * I(\overline{w}) * sv \tag{8}$$

In the same manner for s_2, the lexical-semantic vector lsv_2 is generated by converting all entries of sv to zero, and then executing case-a' and b' as mentioned above.

The cosine coefficient between lsv_1 and lsv_2, as stated in equation 9, is the final value of the semantic sentence similarity.

$$sim(lsv_1, lsv_2) = \frac{lsv_1.lsv_2}{||lsv_1|| * ||lsv_2||} \tag{9}$$

The value of $sim(lsv_1, lsv_2) \in [0, 1]$.

iii. Corpus Statistics:
One can measure the value of distinct words of a sentence using corpus statistics. This is critical because, as indicated in equation 10, one must incorporate stop-words with lower priority relating to other words in a sentence.

$$I(w) = 1 - \frac{Log(x+1)}{Log(S+1)} \tag{10}$$

The frequency of the word w in P is represented by x, while the total number of words in P is represented by S. To prevent zero, x and S are both increased by one. $I(w) \in [0, 1]$.

5. *Cohesion score calculation*:
The cohesiveness score of each sentence in relation to the related document is calculated by determining the Equlidean distance between s_j and the document's centroid dc, as illustrated in equation 11.

$$coh(s_j) = ||(dc - s_j)|| \tag{11}$$

dc calculated using the equation 12.

$$dc = \frac{\sum_{i=1}^{n'} s_i}{n'} \tag{12}$$

Here, $n' \in d_i$ is the number of sentences.

6. *Generating final summary list*
The top m percent sentences based on the *cohesion score* are picked and saved in a new list *NL*, which constitutes the final summary (given in equation 11).

7. *Organising sentences*
An entropy-based mechanism is presented to organise all of the sentences in *NL* according to their relevance (i.e., weight), which is explained below:

- Each word's probability for a sentence is calculated using equation 13.

$$P(w|s) = \frac{term\text{-}frequency(w, s)}{|s|} \tag{13}$$

- Each word's probability for a document d is calculated using equation 14.

$$P(w|d) = \frac{term\text{-}frequency(w, d)}{|d|} \tag{14}$$

The weight of s (referred as $Weight_s$) is determined by its comparison to the document d and is evaluated using equation 16. As illustrated in the equation 15, KL-divergence (*KLD*) (Kumar

[5]determined by experiment

et al., 2009) is used to make the comparison between s and d.

$$KLD(s,d) = \sum_{w} P(w|s) Log(\frac{P(w|s)}{P(w|d)}) \quad (15)$$

$$Weight_s = \frac{1}{KLD(s,d)} \quad (16)$$

Sentences are ordered in the final summary *NL* according to their weights, and constitute the *system-generated summary*.

2.1 *Extractive Gold Summary (EGS) generation*

Sentences containing important information should be categorised as "Important," else they should be labeled as "Not-Important." The sentences identified as' 'Important" are considered for inclusion in the document's summary. The procedures outlined below show how EGS is created from the DUC dataset (P_{duc}) [6].

i) Every document $d \in P_{duc}$ is processed one sentence at a time. For this, the Natural Language Toolkit [7] is employed.

ii) A list L contains all terms of 4 human-written summaries. The number of related words r' between L and s is calculated for each sentence $s \in d$, where r' fluctuates from one sentence to another.

iii) The score for s is measured by the value of r'. Finally, the sentences are ranked and placed in a new list L' depending on these scores.

iv) The extractive gold summary of d is generated by selecting the top m words from L'. The value of m is used to conduct the experiment. In this approach, each P_{duc} document received a 5-sentence extractive gold summary.

3 Analysis of Experimental Results

The description of DUC datasets[8] used for experimental purposes are shown in Table 1. Two most popular techniques, such as ROUGE-N and summary readability, are used to compare the proposed approach with the state-of-the-art approaches, and those are discussed in the following sections.

[6]http://www.duc.nist.gov
[7]http://www.nltk.org/
[8]http://www.duc.nist.gov

3.1 Comparing the performances using ROUGE-N score

i. ROUGE-2 and ROUGE-1 scores (shown in Figures 3 and 2) of the propose model using DUC-2002 dataset are compared with 5 conventional text summarization models (TGRAPH(Parveen et al., 2015), ILP(Woodsend and Lapata, 2010), URANK(Wan, 2010), TextRank(Mihalcea and Tarau, 2004), NN_SE(Cheng and Lapata, 2016)).

ii. Similar way, ROUGE-1, ROUGE-2, and ROUGE-SU4 scores (shown in Figures 4 - 6) of the propose approach using DUC-2006 dataset are compared with 6 conventional text summarization approaches (OnModer(Ye et al., 2007), CTMSUM (Yang et al., 2015), TopicalN(Wang et al., 2007), IIITH-Sum(Jagarlamudi et al., 2006), RMSUM (Zhai and Lafferty, 2017), SFU_v36(Melli)).

iii. Results on the DUC-2002 dataset show that both ROUGE-1 and ROUGE-2 scores of the proposed approach are better than conventional approaches.

iv. Results on DUC-2006 dataset shows that the ROUGE-1 and ROUGE-SU4 scores of the proposed approach are better, but for ROUGE-2 score, CTMSUM and IIITH-Sum are better compared to the proposed approach.

v. Overall from the results of both DUC datasets, it can be concluded that the proposed model is either better or comparable with the existing text summarization techniques.

3.2 Comparing the performance using readability of the summary

Readability of summary means how system-generated summary can read and understand by others in a better manner and is affected by many parameters like sentence weight, sentence length, sentence density etc.(Zamanian and Heydari, 2012). For computing the readability of the summary, statistical methods are generally used (Kondru, 2007), and some of the methods are used by the proposed approach (Table 2). When the

Table 1: DUC Datasets

Dataset	Number of sets	Number of documents	Avg. number of sentence per document	Summary Length	Source
DUC-2006	48	1230	32.22	240	AQUAINT
DUC-2002	54	532	34.55	140	TREC-9

Table 2: Methods of summary readability

Method	Formula
Coleman Liau (CL)	5.89 * (characters/words) - 0.3 * (sentences/words) -15.8
Flesch Kincaid Grade Level (FKGL)	0.39 * (words/sentences) + 11.8 * (syllables/words) - 15.59
Automated Readability Indexing (ARI)	3.70 * (characters/words) + 0.4 * (words/sentences) - 20.42

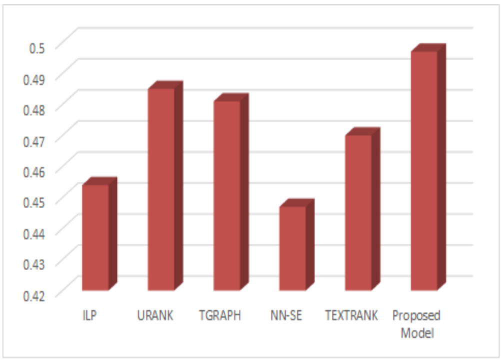


Figure 2: DUC-2002 (ROUGE-1)

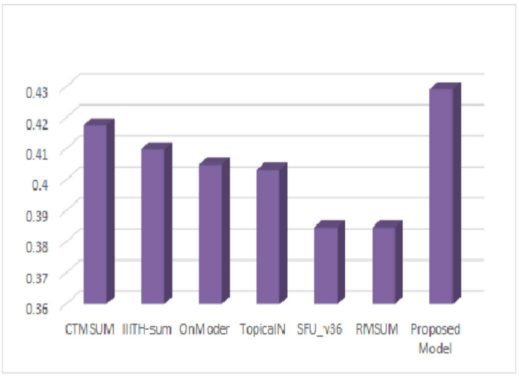


Figure 4: DUC-2006 (ROUGE-1)

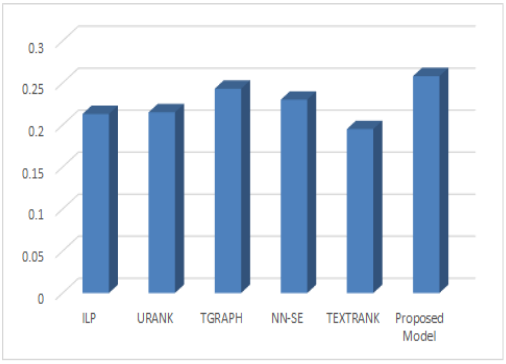


Figure 3: DUC-2002 (ROUGE-2)

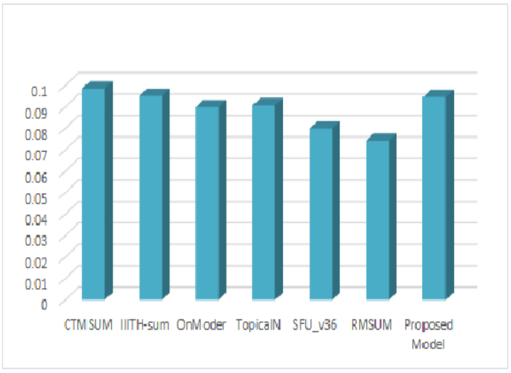


Figure 5: DUC-2006 (ROUGE-2)

summary readability score is very high, it indicates that the system-generated summary is highly user-friendly in terms of understanding and reading. Figures 7 and 8 show the results. Experimentally, it can be concluded that the obtained results of the proposed model are more promising.

4 Conclusion

By combining semantic-based word and sentence similarity, the proposed method suggested a novel

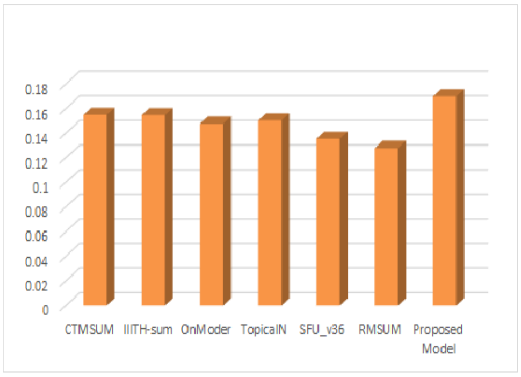


Figure 6: ROUGE-SU4 (DUC-2006)

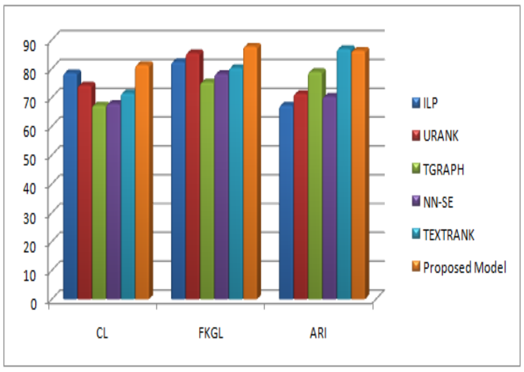


Figure 7: Readbility of summary (DUC-2002)

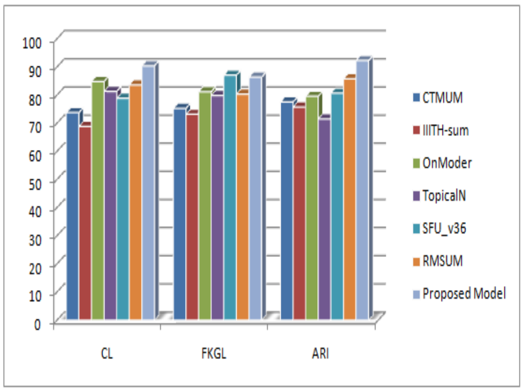


Figure 8: Readbility of summary (DUC-2006)

extractive text summarisation technique. Modified harmonic mean is used to select the important words of each sentence, and then the sentence similarity is computed. Based on the cohesion score, top sentences are selected that constitute the final summary. The sentences in the final summary are organized using KL-divergence approach. The proposed method's experimental work is carried out on two DUC datasets. The proposed approach outperforms the standard approaches on DUC-2006 and DUC-2002 datasets, according to empirical results. This work can be improved even more by using the abstractive text summarization technique to produce a more grammatical-based summary. In the medical domain, many summarization models are proposed, but still, the medical documents have vague terms that make it difficult to extract useful information. The proposed model can use the medical data as the input documents to generate a useful summary that can help the medical system.

References

Asad Abdi, Siti Mariyam Shamsuddin, Shafaatunnur Hasan, and Jalil Piran. 2018. Machine learning-based multi-documents sentiment-oriented summarization using linguistic treatment. *Expert Systems with Applications*, 109:66–85.

Nabil Alami, Mohammed Meknassi, Noureddine Ennahnahi, Yassine El Adlouni, and Ouafae Ammor. 2021. Unsupervised neural networks for automatic arabic text summarization using document clustering and topic modeling. *Expert Systems with Applications*, 172:114652.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, page 484–494.

Reda Elbarougy, Gamal Behery, and AKRAM EL KHATIB. 2020. Graph-based extractive arabic text summarization using multiple morphological analyzers. *Journal of Information Science & Engineering*, 36(2).

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.

Vishal Gupta and Gurpreet Singh Lehal. 2010. A survey of text summarization extractive techniques. *Journal of emerging technologies in web intelligence*, 2(3):258–268.

Jagadeesh Jagarlamudi, Prasad Pingali, and Vasudeva Varma. 2006. Query independent sentence scoring approach to duc 2006. In *In Proceeding of document understanding conference (DUC-2006)*.

Jagadeesh Kondru. 2007. Using part of speech structure of text in the prediction of its readability. *Computer Science & Engineering*.

Chandan Kumar, Prasad Pingali, and Vasudeva Varma. 2009. A light-weight summarizer based on language model with relative entropy. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1752–1753.

Gabor Melli. Description of squash, the sfu question answering summary handler for the duc-2006 summarization task. *safety*, 1:14345754.

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.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Daraksha Parveen, Hans-Martin Ramsl, and Michael Strube. 2015. Topical coherence for graph-based extractive summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1949–1954.

Ted Pedersen, Satanjeev Banerjee, and Siddharth Patwardhan. 2005. Maximizing semantic relatedness to perform word sense disambiguation. *Research Report UMSI 2005/25, University of Minnesota Supercomputing Institute*, 25:2005.

Rajendra Kumar Roul. 2021. Topic modeling combined with classification technique for extractive multi-document text summarization. *Soft Computing*, 25(2):1113–1127.

Rajendra Kumar Roul and Kushagr Arora. 2019. A nifty review to text summarization-based recommendation system for electronic products. *Soft Computing*, 23(24):13183–13204.

Rajendra Kumar Roul, Aditya Bhalla, and Abhishek Srivastava. 2016. Commonality-rarity score computation: a novel feature selection technique using extended feature space of elm for text classification. In *Proceedings of the 8th annual meeting of the Forum on Information Retrieval Evaluation*, pages 37–41.

Rajendra Kumar Roul, Samarth Mehrotra, Yash Pungaliya, and Jajati Keshari Sahoo. 2019. A new automatic multi-document text summarization using topic modeling. In *International conference on distributed computing and internet technology*, pages 212–221. Springer.

Rajendra Kumar Roul, Jajati Keshari Sahoo, and Rohan Goel. 2017. Deep learning in the domain of multi-document text summarization. In *International Conference on Pattern Recognition and Machine Intelligence*, pages 575–581. Springer.

Roger N Shepard. 1987. Toward a universal law of generalization for psychological science. *Science*, 237(4820):1317–1323.

Ladda Suanmali, Naomie Salim, and Mohammed Salem Binwahlan. 2009. Feature-based sentence extraction using fuzzy inference rules. In *2009 International Conference on Signal Processing Systems*, pages 511–515. IEEE.

Xiaojun Wan. 2010. Towards a unified approach to simultaneous single-document and multi-document summarizations. In *Proceedings of the 23rd international conference on computational linguistics*, pages 1137–1145. Association for Computational Linguistics.

Dingding Wang, Shenghuo Zhu, Tao Li, Yun Chi, and Yihong Gong. 2011. Integrating document clustering and multidocument summarization. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(3):1–26.

Xuerui Wang, Andrew McCallum, and Xing Wei. 2007. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 697–702. IEEE.

Kristian Woodsend and Mirella Lapata. 2010. Automatic generation of story highlights. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 565–574. Association for Computational Linguistics.

Guangbing Yang, Dunwei Wen, Nian-Shing Chen, Erkki Sutinen, et al. 2015. A novel contextual topic model for multi-document summarization. *Expert Systems with Applications*, 42(3):1340–1352.

Shiren Ye, Tat-Seng Chua, Min-Yen Kan, and Long Qiu. 2007. Document concept lattice for text understanding and summarization. *Information Processing & Management*, 43(6):1643–1662.

Mostafa Zamanian and Pooneh Heydari. 2012. Readability of texts: State of the art. *Theory & Practice in Language Studies*, 2(1).

Chengxiang Zhai and John Lafferty. 2017. A study of smoothing methods for language models applied to ad hoc information retrieval. In *ACM SIGIR Forum*, volume 51, pages 268–276. ACM.

#covid is war and #vaccine is weapon? COVID-19 metaphors in India

Mohammed Abdul Khaliq, Rohan Joseph, Sunny Rai
School of Engineering Sciences
Mahindra University, Hyderabad, India.
{khaliq170568},{rohan18545}@mechyd.ac.in
sunny.rai@mahindrauniversity.edu.in

Abstract

Metaphors are creative cognitive constructs that are employed in everyday conversation to describe abstract concepts and feelings. Prevalent conceptual metaphors such as WAR, MONSTER, and DARKNESS in COVID-19 online discourse sparked a multi-faceted debate over their efficacy in communication, resultant psychological impact on listeners, and their appropriateness in social discourse. In this work, we investigate metaphors used in discussions around COVID-19 on Indian Twitter. We observe subtle transitions in metaphorical mappings as the pandemic progressed. Our experiments, however, didn't indicate any affective impact of WAR metaphors on the COVID-19 discourse.

1 Introduction

Metaphors are cognitive artefacts to anchor one's thoughts and navigate a situation whether it is social, political or even financial. Conceptual metaphors restructure an abstract domain in terms of a relatively concrete domain, influencing how we perceive reality [Lakoff and Johnson, 1980]. Consider the conceptual metaphor, "DREAMS are BUTTERFLIES". Here, the abstract domain, DREAMS is mapped to a more concrete domain as that of BUTTERFLIES evoking meanings such as *vibrant* and *delicate*. Linguistic metaphors are the manifestations of these conceptual mappings in text. For instance, Her eyes were full of *vibrant* dreams.

India reported the first case of COVID-19 infection in January, 2020 [Andrews et al., 2020]. Soon after that, various control measures including the restriction on international travel, screening of air passengers and institutional quarantine were implemented to curb the infection. The government of India imposed the first nationwide lockdown[1] from Mar 25, 2020 to Apr 14, 2020 as a preventive measure to curb COVID-19.

Various conceptual metaphors with source domains such as WAR (*defeat the virus*), HAMMER/LANDSCAPE (*flatten the curve*) and even MONSTER (*grappling with virus*) were used to communicate state's guidelines as well as reactions to situations arising due to the pandemic [Ruão and Silva, 2021]. In the mapping, COVID-19 is WAR, healthcare staff have been reconceptualised as *warriors*, and the citizens as *soldiers* fighting unitedly against the *enemy* COVID-19. Flusberg et al. [2018] advocate the use of domain WAR to deliver urgent communication infused with motivation. Consider another phrase "*Covid 2.0: Threat of an administrative cytokine storm building up in India*"[2]. Here, administration's RESPONSE is being compared to a biological phenomenon, CYTOKINE STORM to emphasize the lack of attention to ground-reality while drafting COVID-19 protocols. This mapping does bundle subtle aspects such as *failure to identify key points of action*, *overly restrictive protocols* and its resultant *undesired effect on citizens*.

Another direction of research on COVID-19 metaphors reflects on the appropriateness of these cognitive constructs in COVID-19 discourse. There is widespread discontentment against re-imagining the pandemic using source domains such as WAR or MONSTER which may negatively manipulate the understanding of the society. World Emergency COVID-19 Ethics (WeCope) Committee advise against the use of WAR metaphors as it instills fear amongst masses and leads to stigmatization towards those who do not respect the guidelines [WeCope, 2019]. Höijer [2011] calls out the use of metaphors as an instrument by the state to defend its actions and policies during the pandemic. Por-

[1] 'Coronavirus in India: 21-day lockdown begins; key highlights of PM Modi's speech', Business Today (Mar 25, 2020). Available at LINK

[2] Blog by Samir Shukla, Times of India (Mar 30, 2021). Available at https://timesofindia.indiatimes.com/blogs/science-nomad/

Proceedings of the 18th International Conference on Natural Language Processing, pages 431–438
Silchar, India. December 16 - 19, 2021. ©2021 NLP Association of India (NLPAI)

traying the pandemic as a WAR legitimizes state-imposed violence and excessive control. Kahambing [2021] warns against the portrayal of COVID-19 virus as Mother Nature's way to clean the planet. Sabucedo et al. [2020] further demonstrated the ill-effect of commonly used violent source domains such as WAR, MONSTER in COVID-19 discourse on public health. Semino [2021] thus emphasizes the need to reframe COVID-19 metaphors. Rohela et al. [2020] speak in favour of correcting the WAR narrative in COVID-19 discourse using substitute domains such as CRICKET and DANCE. Inspired from this debate, we study the metaphors embedded in Indian tweets posted during the pandemic. The main contributions of this study are:

- We manually identify COVID-19 conceptual metaphors used in headlines of major Indian newspapers published from March'20 to May'21.
- We detect linguistic metaphors embedded in Indian tweets on COVID-19 posted between Mar'2020 to Jul'2021 by fine tuning BERT model.
- Using diachronic embeddings, we detect the transition in manifestations of conceptual metaphors as the pandemic progressed.
- We study the hypothesis if the conceptual metaphor WAR has an affective influence on COVID-19 discussion on Twitter.

The rest of the paper is organized as follows. We discuss the prior works on conceptual metaphors in context of the ongoing global pandemic in Section 2. We describe data collection and the procedure to frame different conceptual metaphors in Section 3. Section 4 discusses the evolving interpretation of mappings as well as the role of WAR metaphors on COVID-19 discourse. We conclude our work in Section 5.

2 Related Work

Wicke and Bolognesi [2020] studied the source domain WAR and how its pervasive nature when describing diseases, plays a role in the COVID-19 dialogue. Prior research discusses the role of conceptual metaphors in moulding public perception in India [Rohela et al., 2020, Wagener, 2020]. Das [2020] describes the crisis and wrath faced by marginalized sections of society due to WAR centered analogies by the Government of India. The readers are encouraged to refer [Rai and Chakraverty, 2020] to know more about metaphors and theories.

Our work differs from the existing works in multiple ways. First, our work focuses on the COVID-19 metaphors of India. We detect metaphorical tweets and also label the underlying conceptual mapping. We study the transition in the linguistic metaphors as the pandemic progressed. Taking inspiration from WEAT [Caliskan et al., 2017], we measure affective influence of metaphorical tweets on COVID-19 discourse as well as see if WAR metaphors indeed present a grimier picture. To the best of our knowledge, this is the first computational approach designed to understand metaphorical themes in India during the pandemic.

3 Discovering Metaphors of COVID-19

3.1 Dataset

Twitter is a micro blogging platform, used widely during the pandemic to express one's feelings and seek help. We extracted Indian tweets on COVID-19 posted between March'20 to July'21 using *snscrape*[3] library. A tweet is considered a COVID-19 tweet if it has at least one COVID-19 related hashtag such as *#coronavirus, #covid19, #quarantine, #covid_19, #vaccine, #TogetherAgainstCovid etc*. To ensure that the extracted tweets are from India, we check if the location attribute within the tweet object pulled by snscrape contains "India". Our dataset comprises of over $1.3M$ tweets.

3.2 Filtering literal tweets

To filter out literal tweets, we fine tuned a BERT model [Devlin et al., 2018]. Two human annotators were asked to tag a random subset of collected tweets into categories *metaphor* and *not metaphor*, for the task of finetuning. Both annotators are undergraduate students aged between $19 - 22$, proficient in English with sufficient knowledge of Indian society. The guidelines shared with annotators to identify metaphors in tweets is as described by Pragglejaz group [Group, 2007]. Below are the steps:

- Read the text to get a general understanding of the meaning
- Determine the lexical units

[3]https://pypi.org/project/snscrape/

  - Establish the contextual meaning of the unit
  - Determine if it has a more basic meaning

- Does the contextual meaning contrast with the basic meaning but can it be understood in comparison with it?
- If yes, mark the unit as metaphorical.

A total of $3.7K$ tweets were tagged by annotators, out of which $1.8K$ were marked as metaphorical. We obtained Cohen's kappa of 0.719 on a common sample of 100 tweets indicating good reliability of annotation. The hand annotated dataset of tweets is available at link[4].

We split this dataset into a training set with 3006 tweets, test and validation sets with 376 tweets each. On finetuning BERT for 25 epochs with a learning rate of $2e^{-5}$, we obtained an accuracy of 74.4% on the validation set. On the test set, we achieved accuracy of 72.6% with a precision of 77% and recall of 67%.

We used this finetuned model to identify metaphorical tweets from the dataset collected in Sec. 3.1. Out of almost $1.3M$ tweets, the system predicted $264K$ tweets as metaphorical. We hereafter indicate this set of metaphorical tweets as $\mathbf{M_0}$.

3.3 Framing the source domain

The next task is to derive a list of conceptual metaphors used in COVID-19 metaphorical tweets from India. #ReframeCovid[5] is one such ongoing open-source work which collects metaphorical mappings present in global COVID-19 tweets and related media.

For our study, we created a list of metaphorical mappings S_0 (of the form SOURCE DOMAIN is TARGET DOMAIN) inspired from #ReframeCovid along with the manual analysis of major Indian newspaper headlines on COVID-19 published during Mar'20-May'21. Few examples are VACCINE is SHIELD, COVID-19 is TEACHER and PANDEMIC is SPEEDBREAKER. The complete list of mappings is available at link[4].

Prior works [Choi and Lee, 2019, Wicke and Bolognesi, 2020] used websites[6] to extract lexical units that they then use to frame the source domains. However, we found that the these lexical units are not used commonly in Indian English. We thus use pretrained *word2vec* embeddings [Mikolov et al., 2013] to expand the set of relatable lexical units/concepts close to a source domain $s \in S_0$. We train a *word2vec* skip-gram [Mikolov et al., 2013] model on the 264K tweets in $\mathbf{M_0}$ to derive the lexical units . We define lower and upper thresholds for cosine similarity to filter out overly specific as well as generic concepts. The thresholds are decided empirically. The set P_s denotes the set of lexical units that we use to frame source domain $s \in S_0$. Below are few lexical units from the set P_{war}:

fight, battle, defeat, enemy, soldier, menace, biological_war, weapon, battlefield, soldier, defend, warfare, bio_war, unseen_enemy, hero, confrontation, army, combat, biowar, invasion, biowarfare, destruction, war, attack, superpower, destruction, fighting, standoff, invisible_enemy, invade, invasion

We manually filter the list to make it contextually suitable for COVID-19 discourse. For instance, the above list for the source domain WAR comprises of words such as *menace*, *hero*, *superpower* which was used in literal sense while discussing the pandemic. Therefore, these words are removed from the list. This forms the basis for identifying underlying source domains in Indian metaphorical tweets.

3.4 Identifying Conceptual Metaphors in Tweets

In this section, we describe our approach to identify the inherent conceptual mapping that is, TARGET DOMAIN is SOURCE DOMAIN in the tweets.

3.4.1 Labeling Source domain

We categorize a tweet t to source domain $s \in S_0$ if t consists a word $w \in P_s$. There is a possibility that a tweet may have words related to two or more source domains. For our analysis, we have eliminated these tweets and will only be utilising tweets that uniquely indicate a particular source domain. Below are few example tweets:

*"A **storm** is coming. Brace yourselves. Impact on Indian economy will be severe. I have started studying and will write a detailed article on it. Will publish soon. #coronavirusindia"*. - COVID-19 is STORM

"In a #World divided by #religion, greed and

[4]https://github.com/makflakes/Covid-Metaphors-of-India/tree/main/data
[5]https://sites.google.com/view/reframecovid/initiative
[6]www.relatedwords.org

Table 1: Top-10 Source Domains

Source Domain	#Tweets
WAR	48415
MONSTER	2884
SUCCESS/CHALLENGE	1382
LESSON/TEACHER	1252
STORM	1213
DARKNESS	1080
PUNISHMENT/BANE	851
PRISON	851
LUXURY	602
CATALYST	542
SAVIOR	486
SHIELD/BARRIER	426

inflated egos, it took an ***invisible*** *virus to instill a common fear. And we still believe that #human beings are the most intelligent species on this planet. #coronavirus"* - COVID-19 is DARKNESS

"Janata Curfew on 22nd March 2020 from 7 AM to 9 PM AND "Ghantanad" on 5 PM, which will help Indians to fight against the corona viruse. It will help to kill the ***devil****"* - COVID-19 is MONSTER

We list the Top-10 source domains on the basis of their volume in $\mathbf{M_0}$ in Table. 1. It may be noted, this list is derived using $s \in S_0$. It is thus possible that there are undetected source domains $s \notin S_0$ with metaphorical tweets in our dataset. From Table. 1, we observe that WAR is the most often used source domain to describe COVID-19 related events. Source domains such as MONSTER, CHALLENGE, LESSON, STORM also contribute significantly to the discourse.

3.4.2 Assumption regarding target domain

Since the tweet extraction process focused only on tweets with COVID-19 related hashtags, it is safe to assume that all tweets are inherently descriptors of COVID-19 and related dialogue. We initially considered segregating tweets with vaccine related hashtags to the target domain VACCINE. We discovered a total of 3701 metaphorical tweets on VACCINE. On careful analysis, we discovered that tweets tagged with VACCINE related hashtags, were also essentially reconceptualizing COVID-19/PANDEMIC. Few such tweets are provided below.

"Another ***deadly*** *wave of Covid19 is ravaging countries including India Stricter observance of anti Covid protocol and stepping up vaccination manifold are urgently called for to face the crisis #tkan #vaccine"* - VIRUS/COVID-19 is MONSTER

"Got vaccinated today with first dose of indigenously developed #Covaxin Thank you narendramodi. Thank you all the scientists who worked hard to invent the vaccine in record time. Together India will ***defeat*** *COVID-19."* - COVID-19 is WAR/VIRUS is ENEMY

Due to the plentiful presence of such tweets in the VACCINE targeted set, we decided to go ahead with COVID-19 as the sole target domain for further analysis.

4 COVID-19 Metaphors of India

4.1 Evolving Conceptual Mappings

As the pandemic progressed, the conceptual mappings describing COVID-19 also evolved. Consider the topic of VACCINE, which was initially conceptualized as a WEAPON[7] to decimate the *enemy*, COVID-19 virus. Later, VACCINE evolved into a PASSPORT/TICKET[8] to freedom which allowed unrestricted movement and gradually, it metamorphosed to LUXURY[9] which was rare and accessible to only few.

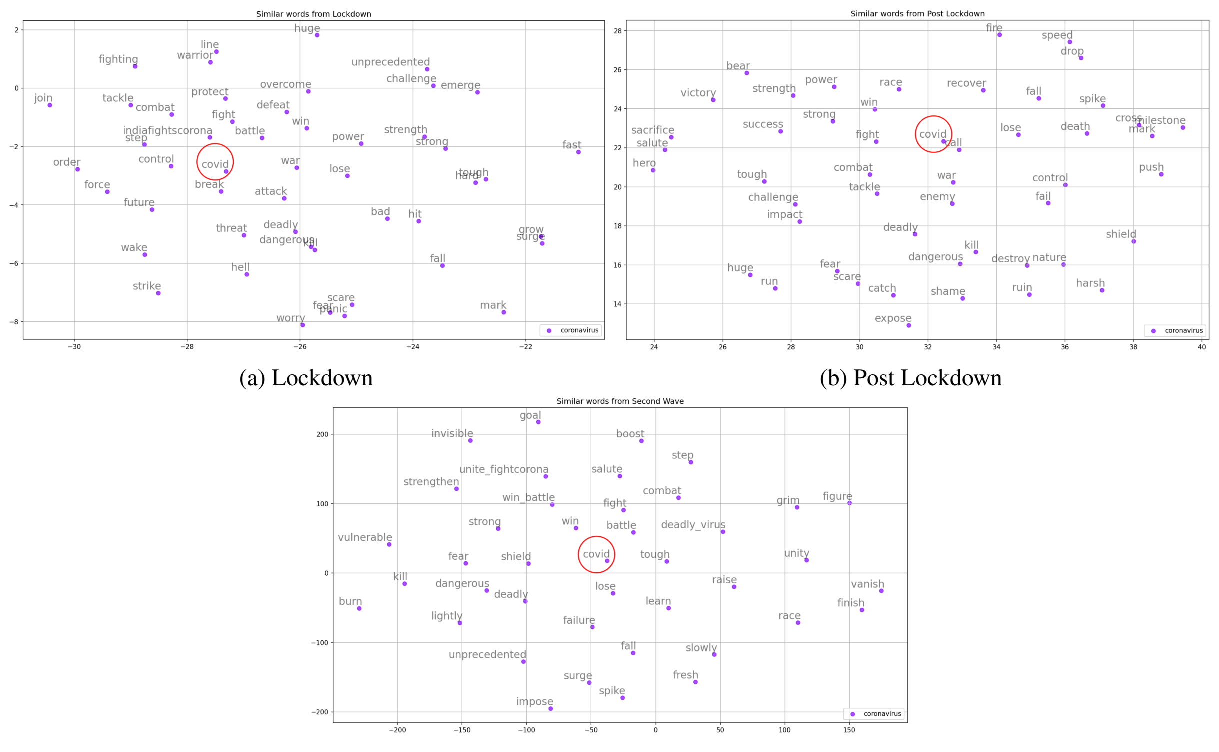
In this section, we study the evolving conceptualization of COVID-19 through the notion of semantic shift. To compute semantic shift, the standard approach is to first slice a corpus with respect to time. The granularity for time slicing may vary depending on the problem statement. For our analysis, we consider the duration (a) t_0: *during lockdown* i.e. Mar'20 to Jun'20 (b) t_1: *post lockdown* i.e. July'20 to Oct'20 and (c) t_2: *second wave* i.e. Mar'21 to Jun'21. Given corpora $\mathbf{C} = [C_{t_0}, C_{t_1}, C_{t_2}]$, the task is to analyse the change if any in semantic neighbourhood of COVID-19. Here, C_{t_i} indicates the set of metaphorical tweets posted during the time interval t_i.

On slicing the set of metaphorical tweets M_0 from Sec. 3.2, C_{t_0} has 136K tweets, C_{t_1} has 39K tweets and C_{t_2} contains 65K tweets. We learn word embeddings using *word2vec* skip-gram architecture [Mikolov et al., 2013] for the first phase using

[7]The Hindu, May 26, 2021. "Vaccination is our only weapon" available at LINK

[8]The Diplomat, July 12, 2021. "Vaccine Passports: Ticket to Freedom or Path to a Divided World? " available at LINK

[9]Quartz India, Aug 2, 2021. "India's vaccine supply is a curious mix of abundance and shortage" available at LINK



(a) Lockdown

(b) Post Lockdown

(c) Second Wave

Figure 1: t-SNE representation of diachronic word embeddings

time-specific corpora C_{t_0}. Here, the vectors are randomly initialized. For next two phases, we update the embeddings initialized with the embeddings learnt from the previous phase. In order to compare word vectors from different time-periods, we align word vectors to the same coordinate axes using orthogonal Procrustes [Hamilton et al., 2016].

4.1.1 Analysis

To analyze the semantic shift in the conceptualization of COVID-19, we plot the t-SNE visualization [van der Maaten and Hinton, 2008] in Fig. 1 for all phases.

Fig. 1-a visualises the semantic concepts related to COVID-19 based on tweets posted in t_0 phase. Fear and determination are reflected from the metaphors such as *scare, fear, panic, worry, deadly, dangerous* and *combat, win, threat defeat, protect* respectively used in this phase. COVID-19 is conceptualized as MONSTER, and even as an OBSTACLE/GAME (*challenge, overcome, strike, tackle*). This overview is consistent with the mixed feelings of fear and hope in the early stages of COVID-19 in India.

The post lockdown embeddings are aligned on t_0 embeddings. We present *covid* related concepts for this period in Fig. 1-b. Metaphors such as *fight, call, win, race* are getting closer to COVID-19 indicating the positive attitude. MONSTER related words such as *fear, scare, deadly* are moving away. BARRIER metaphors such as *shield* and other units such as *nature*, *shame*, *ruin* start to appear in this phase.

Concepts related to *covid* from the Second wave embeddings aligned on t_1 are depicted in Fig. 1-c. Defensive WAR metaphors such as *failure*, *lose*, *tough* and *shield* are getting closer. DARKNESS related metaphors such as *grim*, *vanish*, *invisible* can be seen. MONSTER related metaphors have come closer when compared to Fig. 1-b. Orientational metaphors such as *fall*, and *surge* are also present.

4.2 Linguistic Metaphors

To identify linguistic metaphors of each phase, we extracted the top-1000 words closest to *covid19* from each of these phases. We manually go through this list to identify linguistic metaphorical units. Relevant tweets were retrieved when needed to substantiate the metaphoricity and rule out the literal use. We present these linguistic metaphors in order of increasing distance from *covid19* in Table 2.

The foremost observation is the pronounced presence of WAR metaphors across all phases. This

Table 2: Linguistic Metaphors of COVID-19

Phase	Linguistic metaphors (w.r.t cosine distance from COVID-19 in vector space)
Lockdown (t_0)	fight, indiafightscorona, battle, crisis, deadly, break, war, defeat, support, control, win, combat, attack, force, team, protect, hide, coronawarrior, threat, kill, dangerous, suffer, prepare, impact, isolation, hit, border, hell, solution, strategy, survive, fighting, coronafighter, unite, develop, strength, scary, destroy, isolate, disaster, tackle, scare, beat, deadly_virus weapon, soldier, fighter, shut, shame, lesson, cover, enemy, scared, win_battle, win_war, danger, devastating, catch, surge, tracking, deep, victory, vulnerable, evil, cut, surpass, giant, expose, break_chain, boom, unlock, push, hail, chain, flatten_curve, heal
Post-Lockdown (t_1)	indiafightscorona, break, stand, block_case, battle, line, strong, support, war, combat, save, warrior, frontline, duty, fighter, leader, play, win, strategy, fear, hit, control, covidwarrior, scary, deadly, attack, base, push, tackle, brave, hell, team, force, defeat, game, fall, race, power, action, united, build, struggle, beat, dangerous, kill, strike, panic, powerful, peak, danger, stage, crusader, frontline_warrior, tough_time, scared, deadly_virus, deep, crisis, hide, throw, win_battle, lose_life, rage, threat, grim, nightmare, block, havoc, unlock, fire, fighting, flood, wall, victory, impact, kick, boost, storm, invisible, weapon, disaster, shoot
Second Wave (t_2)	battle, support, deadly, crisis, strong, win, suffer, defeat, hit, indiafightscovid, handle, dangerous, safety, warrior, control, strength, protect, overcome, fear, difficult, attack, tough_time, hard, scary, war, kill, struggle, win_battle, tough, scare, pain, beat, peak, panic, strike, grim, catastrophe, save_life, unite_fightcorona, hell, shame, difficult_time, lose_life, combat, frontline_worker, stay_united, devastating, disaster, wake, hero, breakthechain, powerful, shift, destroy, fighting, deadly_virus, indiafightsback, blast, tackle, seek, shocking, dip, weapon, force, rule, frontline_warrior, game, chance, strategy, decline, hang, target, lightly, enemy, border, wish_speedy, lose_battle, threat, loss, shall_pass, push, catch, build, breach, blame, player, rage, bio_bubble, shock, frontline, hit_hard, nightmare, gloom, danger, tsunami, tear, kick, casualty, terrible, brutal, lethal, mark, pressure, devastate, devastation,

includes *enemy, fight, defeat, attack, weapon, battle, casualty* etc. We further note the presence of domains including MONSTER (*deadly, scary, giant, fear*), GAME (*team, race, tackle, push, player, strike*), STORM/DISASTER (*crisis, devastate, havoc, flood, hail, tsunami*), DARKNESS (*hide, cover, nightmare, gloom, grim, invisible*), HAMMER (*beat, flatten_curve, hit, impact*) and ACCIDENT (*chance, rage, shock*). There are also other metaphors such as *dip, surge, blast, tear, kick, build, shame, hell, evil, heal, chain, deep* in the corpus.

During t_0 phase, the closest words are *fight, battle, crisis, deadly, break, war, protect, hide* etc. It may be noted that India had only few reported cases of COVID-19 in comparison to t_1 phase. Nevertheless, the metaphors are relatively grim and fear inducing. The fear of unknown and the lack of confidence on Indian healthcare might be the reason behind these overly gloomy tweets.

We note increased use of GAME metaphors in t_1 phase when compared to other two phases. It may be noted that Post-Lockdown is the phase where India faced the first wave of COVID-19. COVID-19 is discussed using concepts such as *race*, *action*, *kick*, *team* and *block*. Moreover, even WAR metaphors are used in more authoritative fashion when compared with the Second Wave phase. This indicates the transition in lexical manifestations of WAR metaphors while discussing COVID-19. Metaphors such as *duty, strategy, push, tackle, brave, fighter, crusader* convey a sense of control. The metaphors used in this phase indicate a more confident and controlled reaction to the pandemic in comparison to t_0 and t_2 phases.

We see the highest volume of metaphors in *Second Wave* phase. There are more negative metaphors such as *battle, deadly, crisis, suffer, defeat, dangerous* closer to COVID-19 when compared with the previous two phases. There are also increased occurrence of DISASTER/DARKNESS metaphors such as *grim, catastrophe, devastating, disaster, panic, nightmare, gloom, danger, tsunami* etc. We also note metaphors such as *breach, blame, rage, pressure* from GAME domain indicating the shift in meaning of GAME metaphors. There is a clear difference in the underlying emotional tone of metaphors when compared with t_1 phase. The first wave (t_1) definitely saw a controlled strategy with manageable COVID-19 cases whereas the second wave (t_2) witnessed more suffering, panic and lack of control which is also evident from the metaphors.

4.3 Impact of WAR metaphors on COVID-19 online discourse

To better understand the role of WAR metaphors in COVID-19 discourse, we analyse if metaphors based on WAR mapping indeed paint an overly grim picture contrary to the true reality of COVID-19 in India. We take inspiration from Word Embedding Association Test (WEAT) proposed by Caliskan et al. [2017] to identify the polarity of associativity between WAR metaphors and COVID-19 concepts. We define our hypothesis $\mathcal{H}_a$ as " WAR has an affective influence if COVID-19 concepts show associativity towards negative or positive attribute sets." The null hypothesis $\mathcal{H}_o$ therefore is " WAR metaphors have no affective influence on the meaning of COVID-19 concepts."

Let $\mathcal{D}_o$ be the dataset of non-metaphorical Indian tweets as predicted by the fine-tuned BERT model in Sec. 3.2. Let $\mathcal{D}_s$ be the set of predicted metaphorical tweets belonging to source domain $s \in S_0$. For our analysis, we first learn word embeddings using skip-gram word2vec model on the dataset $\mathcal{D}_o$. These representations will serve as the baseline for our analysis. We fine tune the learnt embeddings using $\mathcal{D}_s$ to capture if there is a change in the meaning of COVID-19 concepts due to the source domain s.

For analysis, let $\mathbf{X}$ be the set of COVID-19 target words, and $\mathbf{P}$, $\mathbf{N}$ be the attribute sets namely *positive* and *negative* respectively. We define $\delta(\mathbf{X}, \mathbf{P}, \mathbf{N})$ as the differential association of the target words embeddings for $x \in \mathbf{X}$ trained on $\mathcal{D}_o$ and $\mathcal{D}_o + \mathcal{D}_s$ with the attribute sets $\mathbf{P}$ and $\mathbf{N}$ as in eq. 1.

$$\delta(\mathbf{X}, \mathbf{P}, \mathbf{N}) = \sum_{x \in X} f(\vec{x}_o, \mathbf{P}, \mathbf{N}) - \sum_{x \in X} f(\vec{x}_s, \mathbf{P}, \mathbf{N}) \tag{1}$$

where

$$f(x, \mathbf{P}, \mathbf{N}) = \mu_{p \in \mathbf{P}} \cos(x, p) - \mu_{n \in \mathbf{N}} \cos(x, n)$$

- $\vec{x}_o$ refers to the embeddings for $x \in \mathbf{X}$ learnt on $\mathcal{D}_o$
- $\vec{x}_s$ refers to the embeddings for $x \in \mathbf{X}$ fine tuned on $\mathcal{D}_s$
- μ indicates mean,
- $cos(\vec{a}, \vec{b})$ denotes the cosine similarity between the vectors $\vec{a}$ and $\vec{b}$.

Each word in sets $\mathbf{X}, \mathbf{P}$ and $\mathbf{N}$ has occurred at least 20 times in both corpus and are provided below:

$\mathbf{X}$ = { covid, corona, virus, lockdown, pandemic, coronavirus, health, hospital }

$\mathbf{P}$ = {hope, faith, strength, unite, support, care, survive, recover}

$\mathbf{N}$ = {death, panic, struggle, concern, stress, chaos, shortage, oxygen}

Using permutation test for sampling, we discovered that the domain WAR is more close to attribute set $\mathbf{P}$ with p-value of 0.98. We reject our hypothesis $\mathcal{H}_a$ due to high p-value. It is thus not evident from our experiments that WAR metaphors have statistically significant affective influence on COVID-19 domain. We further performed this analysis specifically for tweets posted during t_2 phase. However, our experiments did not reveal any affective influence exercised due to metaphors on the understanding of COVID-19. In future, we aim to design extensive experiments to understand the affective influence of different metaphorical themes on COVID-19 discourse.

5 Conclusion

Collecting data for any figurative text related task is a big challenge. Through this study, we release a hand-annotated set of $3.7K$ Indian tweets for metaphor related research. A wide variety of conceptual mappings were used in Indian newspapers while reporting COVID-19 situation in India. Nevertheless, we see a handful of these conceptual mappings in Indian tweets. WAR, MONSTER, DARKNESS and GAME are the most prominent conceptual metaphors in Indian tweets. Our results reveal the shift in the use of conceptual metaphors as the pandemic progressed. Despite intense discussions on the appropriateness of the conceptual mappings used during the pandemic, it is not evident from our experiments if WAR indeed led to an overly negative understanding of COVID-19 in India.

References

MA Andrews, Binu Areekal, KR Rajesh, Jijith Krishnan, R Suryakala, Biju Krishnan, CP Muraly, and PV Santhosh. First confirmed case of covid-19 infection in india: A case report. *The Indian Journal of Medical Research*, 151(5):490, 2020.

Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186, 2017.

Won-Gyu Choi and Kyung-Soon Lee. Conceptual representation for crisis-related tweet classification. 2019.

Shreshtha Das. India's war on covid-19: how the government is turning marginalised citizens into suspected enemies and criminals. *South Asia@ LSE*, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Stephen J Flusberg, Teenie Matlock, and Paul H Thibodeau. War metaphors in public discourse. *Metaphor and Symbol*, 33(1):1–18, 2018.

Pragglejaz Group. Mip: A method for identifying metaphorically used words in discourse. *Metaphor and symbol*, 22(1):1–39, 2007.

William L Hamilton, Jure Leskovec, and Dan Jurafsky. Diachronic word embeddings reveal statistical laws of semantic change. *arXiv preprint arXiv:1605.09096*, 2016.

Birgitta Höijer. Social representations theory: A new theory for media research. *Nordicom review*, 32(2): 3–16, 2011.

Jan Gresil S Kahambing. Metonymies, metaphors and/or language reconsiderations for sustainability during covid-19. *Journal of Public Health (Oxford, England)*, 2021.

George Lakoff and Mark Johnson. Conceptual metaphor in everyday language. *The journal of Philosophy*, 77(8):453–486, 1980.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Sunny Rai and Shampa Chakraverty. A survey on computational metaphor processing. *ACM Computing Surveys (CSUR)*, 53(2):1–37, 2020.

Pallavi Rohela, Anant Bhan, Divya Ravindranath, Devi Leena Bose, and Soumitra Pathare. Must there be a "war" against coronavirus. *Indian J Med Ethics*, pages 222–226, 2020.

Teresa Ruão and Sónia Cristina Melo Silva. Strategic science communication: the "flatten the curve" metaphor in covid-19 public risk messaging. 2021.

José-Manuel Sabucedo, Mónica Alzate, and Domenico Hur. Covid-19 and the metaphor of war (covid-19 y la metáfora de la guerra). *International Journal of Social Psychology*, 35(3):618–624, 2020.

Elena Semino. "not soldiers but fire-fighters"–metaphors and covid-19. *Health Communication*, 36(1):50–58, 2021.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

Albin Wagener. Crushed by the wheels of industry: War, heroes, and domestic recolonization in the time of covid-19. *Postdigital Science and Education*, 2: 576–580, 2020.

World Emergency COVID19 Pandemic Ethics Committee WeCope. A call to cease the use of war metaphors in the covid-19 pandemic. 2019. URL https://eubios.info/yahoo_site_admin/assets/docs/WeCope_Call_to_Cease_the_War_Metaphor.167174242.pdf.

Philipp Wicke and Marianna M. Bolognesi. Framing covid-19: How we conceptualize and discuss the pandemic on twitter. 2020. doi: 10.1371/journal.pone.0240010. URL https://doi.org/10.1371/journal.pone.0240010.

Studies Towards Language Independent Fake News Detection

Soumayan Bandhu Majumder[1] and **Dipankar Das**[1]

[1]Computer Sc. & Engineering. Department, Jadavpur University
{soumayanmajumder, dipankar.dipnil2005}@gmail.com

Abstract

Fake news, one of the important topics of recent trends causes serious problems to common people and even organizations in general by spreading its threads in terms of news and social messages. The scenario becomes vulnerable while we deal with health issues like COVID19. Thus, in the present task, we have collected the tweet data on COVID19 of seven different languages. We employed two types of model, one works in a language dependent way whereas the other one aims to investigate various language independent issues. We received better results in language independent model for the languages like English, Hindi and Bengali. Results of European languages like German, Italian, French and Spanish are comparable in both language dependent and independent models.

1 Introduction

In our day-to-day world, we humans mostly generate unstructured data and mostly the textual data in a large scale. While utilizing information grounded in such textual data, our efforts become useless when we stuck in handling fake news or misinformation. We observe three main different types of news – legitimate news, fake news and satirical news and all these news differ with respect to two parameters; authenticity and intent. If authenticity of news is not verifiable or false and its intent is to mislead the readers, then it is known as fake news. On the other hand, if authenticity of the news is verifiable or true and its intent is to convey or spread the authenticity to the readers then it is known as legitimate news. Finally, if authenticity of news is not verifiable or false and its intent is oriented towards entertainment, then such type of news is known as satirical news.

In the present approach, we have considered only fake and non-fake news. Here we use linguistic features for detecting fake news. We tried to detect fake news in language dependent and independent both ways. We have also checked which features are more important than others in detecting fake news for a particular language.

Our remaining paper is divided into different sections. In Section 2 we will see some studies or previous works related to fake news detection. In Section 3 we will see our dataset, upon which we performed this research work. After that in next Section we will discuss methods or model architecture. In Section 5 we discussed about result and error analysis and lastly in Section 6 we draw a conclusion and discussed about future works.

2 Related Work

Misinformation is currently one of the balmy topics of last six to seven years. In this Fake news field particularly many researches are already been executed and many are currently also going. Researchers suggested many different ways of detecting fake news. If we generalize them, then we can come to a conclusion that we can detect a news fake or not based news content or social context. These are the only two generalize way of detecting fake news. We can call these two ways – a) news content model, and b) social context model. Again there are two divisions of news content model based upon which we can detect a fake news – a) knowledge based detection, and b) style based detection. For knowledge based method everything is depend on a knowledge base that we extracted from the news. After creating that knowledge base we have to compare it with some reliable source to check its authenticity. In style based method we mainly focus on linguistic features and based on that we predict the news. Like news content model social context model also divided in two categories, based on which we can detect a fake news – a) propagation based techniques, and b) credibility based techniques. In case of propagation based method, we have to find propagation path of the news on the social media and have to track the original source of the news. But for credibility based method, we have to find the various relationship between news article and users, publishers, posts, shares, comments etc.

Several researchers have explored the area into different ways. George et. al. uses different types of machine learning algorithms like SVM, Naïve Bayes, KNN etc. upon contextual features and linguistic features to detect fake news. In contrast, Perez-Rosas et. al. analyse seven different types of news domains and also analyse their linguistic differences in both fake and neutral news and also compare characteristics of different domains. Whereas Bedi et. al. uses knowledge based fake news detection mechanism. He creates a knowledge database first and then compare it to authorized news database to verify fake news and neutral news. Dey et. al. follows style based detection. So he first extracted features and then analyse the linguistic patterns and then apply KNN algorithm to classify news. Uppal et. al. propose discourse level analysis for deception detection of news documents.

However, these above mentioned techniques have one problem that is they can detect the fake news of a particular language and particular domain. But, one of the important issues is that whether we can detect it in a language independent fashion or not. Therefore, in the present attempt, one of our aims is to detect the fake news in language independent way. Moreover, none of the above mentioned approaches deal with less computerized and less resourced language like Bengali. Here, in the present task, we have developed dataset as well as explored the detection techniques for Bengali along with English, Hindi and other European languages.

3 Dataset Preparation

Undoubtedly, the term fake news comes into our mind while we think of social media. Thus, we aimed to collect data from a social media like twitter. We also collected newspaper data from different languages like English, Hindi and Bengali.

Already there is a popular multilingual fake news dataset present in covid19 domain, but this dataset does not contain German and Bengali languages and secondly size of our dataset is much larger than this dataset. In case of newspaper data, we crawled sentences (mostly news) from various web sources and manually labeled them. We crawled Bengali sentences from 'ABP Ananda[1]' and Hindi sentences from 'Abp News[2]' and 'Aajtak[3]', as well as 'Twitter[4]'.

In case of collecting data for European languages, we considered the data available in the CLEF shared task by participation.

In addition, we have collected COVID19 twitter data from 15th March, 2020 to 15th May 2020 of 7 different languages (German, Italian, French, Spanish, English, Hindi and Bengali). We collect our COVID19 related data from twitter using tweet-scrapper library. It has been observed in preliminary study that fake news datasets are always skewed because the frequency of real news data are much more than fake news data. Thus, we tried to maintain a similar ratio of fake and real news in each of the languages. However, we were able to collect very less number of fake tweets in Bengali and Hindi in COVID19 domain. Thus, in order to train the models for Bengali and Hindi, we had to add more data from other domain as well.

When we started collecting our dataset from twitter, we have the following tweet filtering feature options like '*tweet_id*', *'hashtags'*, *'has_media'*, *'is_replied'*, *'is_reply_to'*, '*img_urls*', *'links'*, *'likes'*, *'parent_tweet_id'*, *'replies'*, *'reply_to_users'*, *'retweets'*, *'screen_name'*, *'text'*, *'text_html'*, *'timestamp'*, *'timestamp_epochs'*, *'tweet_url'*, *'user_id'*, *'username'* and *'video_url'*.

But among these feature columns, we used only the '*text*' column and extracted textual features which we employed in our language dependent model. Some of such textual features are *'label'*, *'word_count'*, *'char_count'*, *'word_density'*, *'punctuation_count'*, *'title_word_count'*, *'upper_case_word_count'*,

[1] https://bengali.abplive.com/

[2] https://www.abplive.com/

[3] https://aajtak.intoday.in/

[4] https://twitter.com/?lang=bn/hn

'noun_count', 'adj_count', 'verb_count', 'pron_count', 'adv_count', 'other_POS', 'sentiment', 'tags', 'tags_ORG', 'tags_PER', 'tags_LOC', 'tags_MISC'. For annotating our dataset, we considered the help of factcheck.com. The detail statistics of the dataset are shown in Table 1.

Language	Real # Sentence	Fake # Sentence
German	14155	302
Italian	13270	507
French	13318	300
Spanish	12113	496
English	11490	2097
Bengali	1051	449
Hindi	615	287

Table 1: Number of tweet sentences available for each language

4 Language Dependent Classification

In order to investigate the roles of language to detect fake news, here each of the models and its input features are exclusive to that particular language. We employ different types of machine learning algorithms and also ensemble them in order to achieve better results by exploring the benefits of individual machine learning classifiers.

Here we first extracted some features from the tweets. We here don't use twitter specific features because we want our experiment to be on general purpose, instead of twitter specific fake news. Therefore, we use different types of open source libraries to extract different types of features from the text. Here, we have mainly conducted experiment upon 7 different languages. Among these 7 languages we collected our COVID19 related data purely in 5 languages and for other two languages (Bengali and Hindi), we added data from other domain as the COVID19 data of these two languages were very less in number.

From each text or tweet, we extracted a couple of features for different languages using different libraries. For English and European languages, we used spacy[5] and polyglot library[6]. Like for French we have to download '*fr_core_news_sm*' (which is exclusively for French). Finally, we had to install spacytextblob library[7]. For Hindi language, we use nltk library[8] and for Bengali language, we use Bengali-NLP library[9].

We extracted 17 different features (e.g., *'word count', 'char count', 'word density', 'punctuation count', 'title word count', 'upper case word count', 'noun count', 'verb count', 'adjective count', 'pronoun count', 'adverb count', 'other POS', 'sentiment', 'tags_LOC', 'tags_MISC', 'tags_PER', 'tags_ORG'*).

Classification algorithms are of three types, such as - binary classification, multiclass classification and multi-label classification. Here, we have used binary classification algorithms, because our output is between any one of the fake and real class. We have used Logistic Regression, KNN, SVM, Random Forest, XGBOOST, Ensemble Learning and Naïve Bayes to accomplish our goals. Here we take logistic regression based model as our baseline model.

4.1 Feature Ablation Study

In order to identify the importance of different features for different languages we use random forest algorithm (for entropy) and we also use correlation matrix for checking collinearity between features. Followings are some of the hints into that direction.

English Language: For English language, we checked up to the top 15 important features. Here *sentiment* feature is the most important (more than 0.3) and *char count*, *word density* features are the next to it (close to 0.05) and others are of very less important (less than 0.05).

German Language: Here, the most important feature is *sentiment* (more than 0.6) and next to it is *title word count* feature (close to 0.1), which is very less important than sentiment and other features (less than 0.05) are of very negligible importance.

Italian Language: For Italian language, the most important feature is *sentiment* (more than 0.6) and next to it is *miscellaneous tag* feature (less than

[5] https://spacy.io/

[6] https://polyglot.readthedocs.io/en/latest/

[7] https://pypi.org/project/spacytextblob/

[8] https://www.nltk.org/

[9] https://github.com/sagorbrur/Bengali-NLP-Library

0.05) which is of very less important than the first one and other features are of very less importance.

Spanish Language: Here, we can say that *sentiment* is the most important feature (more than 0.8) and other features importance (less than 0.05) is close to 0, or we can say they are given no importance.

French Language: For French language, *sentiment* is the most important feature (more than 0.45) and then the *adjective count* feature (close to 0.05). This is also of very less important and other features (less than 0.05) are of negligible importance.

Hindi Language: For Hindi language, we can see none of the features are that important because all features have values less than 0.1. But among these, '*char count*' has the highest importance which is slightly greater than 0.07.

Bengali Language: In Bengali language, *noun count* feature has the highest importance (value greater than 0.2) and other POS feature has also some importance (value 0.1). *Char count*, *punctuation and pronoun count* have some importance but those are very less.

4.2 Results

It was noticed that our data is highly imbalanced so accuracy should not be a good metric or score to measure the performance of the models or even to compare the models. Thus, we tried different types of scores like precision, recall, f1 score for every class and macro f1 score for both the classes as a whole. We also calculated AUC (*Area Under Curve*) for each model in each language to do a better comparison among the various models.

Logistic Regression: In logistic regression model for real labelled data, the highest precision achieved is 0.99 for German and Spanish languages and the highest recall is 0.99 for German, Italian, Spanish, French and the highest F1 score is 0.99 for German, Italian and Spanish. In case of fake labelled data, our logistic regression achieved the highest precision of 0.83 for German language and the highest recall of 0.75 for Spanish and the highest F1 score of 0.79 for Spanish. Overall, in case of both real and fake data consideration, we can say Spanish language gives the best result according to both Macro F1 and AUC score.

KNN: In KNN model for real labelled data, the highest precision is 0.99 for German, Italian, Spanish, French language and the highest recall is 0.99 for German, Italian, Spanish and the highest F1 score is 0.99 for German, Italian, French and Spanish. In contrast, for fake labelled data, the highest precision is 0.89 for Spanish language and the highest recall is 0.92 for German, French and the highest F1 score is 0.88 for Italian. As a whole, by taking both real and fake data into consideration, German language gives the best result according to both Macro F1 and AUC score.

SVM: Similarly, in SVM model for real labelled data, the highest precision, recall and F1 scores are 0.99, 0.99 and 0.99 respectively for German, Italian, Spanish, French languages. In SVM model for fake labelled data, the highest precision is 0.91 for German language and the highest recall is 0.97 for Spanish and the highest F1 score is 0.92 for German. Overall, German language gives the best result according to Macro F1 and Spanish gives the best result according to AUC score.

Random Forest: In random forest model for real labelled data, the highest precision, recall and F1 scores were obtained for German, Italian, French and Spanish whereas for fake, the highest precision is 0.93 for German language and highest recall is 0.97 for Italian and the highest F1 score is 0.94 for German and Italian both. It was found that German, Italian language gives the best result according to Macro F1 score and Italian language gives the best result according to AUC score.

XGBOOST: In XGBOOST model for real labelled data, the highest precision, recall and F1 scores were obtained for German, Italian, French and Spanish. In XGBOOST model for fake labelled data, the highest precision is 0.90 for German language and the highest recall is 0.96 for Italian and the highest F1 score is 0.91 for German, Spanish. Overall, Spanish language gives the best result according to Macro F1 and Italian language gives best result according to AUC score.

Stacked Ensemble: Similarly, in stacked ensemble model for real labelled data, the highest precision recall and F1 scores were obtained of 0.99 for German, Italian and Spanish. In stacked ensemble model for fake labelled data, the highest precision is 0.92 for German language and the highest recall is 0.96 for Spanish and highest F1 score is 0.93 for German. Finally, German language gives the best result according to Macro F1 and Spanish gives the best result according to AUC score.

Naïve Bayes: In Naïve Bayes model for real labelled data, the highest precision is 0.99 for German, Italian, Spanish, French language and the highest recall is 0.99 for Spanish, Italian and the highest F1 score is 0.99 for Italian and Spanish. In Naïve Bayes model for fake labelled data highest precision is 0.82 for Italian, Spanish language and the highest recall is 0.93 for Spanish and highest F1 score is 0.87 for Spanish. Spanish language gives best result according to both Macro F1 and AUC score on both the classes.

4.3 Error Analysis

Logistic Regression: In this section first we have to know two things Type-1 error and Type-2 error. Type-1 error is false positive and Type-2 error is false negative. For rest of the paper I will use T1 error for Type-1 error and T2 error for Type-2 error.

Our main concern should be fake news because we don't want to left out any of the fake news as real news. So we need basically recall value of fake labelled data. Here Spanish have highest recall of 0.752 for fake label. So we can say for logistic regression model Spanish language has least error in detecting fake news.

KNN: For English language T1 error is 71 and T2 error is 424. In German language T1 error is 13 and T2 error is 7. For Italian language T1 error is 19 and T2 error is 14. For Spanish language T1 error is 13 and T2 error is 15. For French language T1 error is 36 and T2 error is 76. For Hindi language T1 error is 15 and T2 error is 38. For Bengali language T1 error is 22 and T2 error is 116.

Our main concern should be T2 error because we don't want to left out any of the fake news as real news. So we need basically recall of fake labelled data to calculate T2 error in relative way. Here German have highest recall of 0.92 for fake label. So we can say it has least relative Type-2 error.

SVM: For English language T1 error is 126 and T2 error is 304. In German language T1 error is 5 and T2 error is 16. For Italian language T1 error is 18 and T2 error is 6. For Spanish language T1 error is 14 and T2 error is 5. For French language T1 error is 29 and T2 error is 15. For Hindi language T1 error is 13 and T2 error is 31. For Bengali language T1 error is 15 and T2 error is 54.

Here Spanish have highest recall of 0.966 for fake label. So we can say it has least relative T2 error.

Random Forest: For English language T1 error is 74 and T2 error is 223. In German language T1 error is 7 and T2 error is 4. For Italian language T1 error is 15 and T2 error is 4. For Spanish language T1 error is 19 and T2 error is 8. For French language T1 error is 14 and T2 error is 8. For Hindi language T1 error is 15 and T2 error is 31. For Bengali language T1 error is 28 and T2 error is 98.

Here Italian language have highest recall of 0.97 for fake label. So we can say it has least relative T2 error.

XGBOOST: For English language T1 error is 116 and T2 error is 199. In German language T1 error is 9 and T2 error is 7. For Italian language T1 error is 23 and T2 error is 6. For Spanish language T1 error is 14 and T2 error is 8. For French language T1 error is 19 and T2 error is 19. For Hindi language T1 error is 14 and T2 error is 32. For Bengali language T1 error is 36 and T2 error is 89. Here Italian language have highest recall of 0.96 for fake label. So we can say it has least relative T2 error.

Stacked ensemble model: For English language T1 error is 109 and T2 error is 249. In German language T1 error is 7 and T2 error is 6. For Italian language T1 error is 17 and T2 error is 12. For Spanish language T1 error is 19 and T2 error is 5. For French language T1 error is 19 and T2 error is 20. For Hindi language T1 error is 11 and T2 error is 37. For Bengali language T1 error is 17 and T2 error is 118.

Here Italian language have highest recall of 0.96 for fake label. So we can say it has least relative T2 error.

Naïve Bayes: For English language T1 error is 973 and T2 error is 161. In German language T1 error is 96 and T2 error is 14. For Italian language T1 error is 29 and T2 error is 16. For Spanish language T1 error is 30 and T2 error is 10. For French language T1 error is 83 and T2 error is 18. For Hindi language T1 error is 38 and Ty2 error is 18. For Bengali language T1 error is 53 and T2 error is 31.

Here Spanish language have highest recall of 0.93 for fake label. So we can say it has least relative T2 error.

5 Language Independent Classification

Here, we have conducted experiments to see if we can detect fake news in language independent way or not. We here mainly focused on multilingual BERT model. This BERT model is already pre-trained on some different corpus. Therefore, we will first fine tune this multilingual BERT model with our different language corpora, then we received a fixed length embedded output through this model for each of the tweets, then we pass this output through the two layers of artificial neuron network of different unit size and at last we pass that output through the sigmoid layer.

5.1 Model Architecture

Here, we discuss about our independent language model architecture. First, in pre-processing step, we remove all urls and html tags. Then, we remove all emojis and emoticons present in the tweet. Now, we send these pre-processed tweets into BERT model as mentioned in the input format section of this thesis. After that, we choose to go with pooled output and then passed it through the 256 RELU units of artificial neural network (ANN) layer. Finally, we introduce dropout of 0.4. After that, we passed that through the 128 RELU units and lastly through the sigmoid unit which will give our ultimate output.

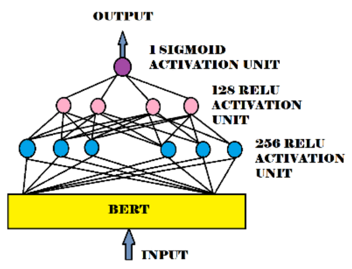


Figure1: Independent model architecture

5.2 Result

In our language independent model for real labelled data, the highest precision is 0.99 for English, Italian, Spanish, French language and the highest recall is 100% for German and the highest F1 score is 0.99 for English, German, Italian, French and Spanish.

In language independent model for fake labelled data highest precision is 100% for German language and the highest recall is 0.98 for French and the highest F1 score is 0.98 for German. Overall, by considering both the real and fake data, we can conclude that English and German language give the best result in Macro F1 score.

Langua		F1	Macro F1
English	Real	0.99	0.98
	Fake	0.97	
German	Real	0.99	0.98
	Fake	0.98	
Italian	Real	0.99	0.93
	Fake	0.88	
Spanish	Real	0.99	0.97
	Fake	0.95	
French	Real	0.99	0.94
	Fake	0.90	
Hindi	Real	0.87	0.78
	Fake	0.71	
Bengali	Real	0.86	0.68
	Fake	0.56	

Table 2: Results of Language Independent Models

5.3 Error Analysis

For English language, T1 error is 11 and T2 error is 17. In German language T1 error is 0 and T2 error is 3. For Italian language T1 error is 22 and

T2 error is 13. For Spanish language T1 error is 5 and T2 error is 7. For French language T1 error is 22 and T2 error is 2. For Hindi language T1 error is 16 and T2 error is 17. For Bengali language T1 error is 30 and T2 error is 69.

Our main concern should be T2 (Type-2) error because we don't want to left out any of the fake news as real news. So we need basically recall of fake labelled data to calculate T2 error in relative way. Here, in case of French language, the highest recall of 0.98 for fake label. So we can say it has least relative T2 error.

6 Result Comparison

After doing both language dependent and independent fake news detection, now we will compare both results.

Languages	Language Dependent		Language Independent	
	F1 score	Recall	F1 score	Recall
English	0.83	0.69	0.98	0.97
French	0.93	0.92	0.94	0.98
German	0.96	0.95	0.98	0.96
Italian	0.96	0.97	0.93	0.91
Spanish	0.95	0.97	0.97	0.95
Hindi	0.66	0.69	0.78	0.71
Bengali	0.67	0.66	0.68	0.48

Table3: result comparison between two models

Here, we compare our results based upon two parameters. Firstly, for all over performance we take F1 score as our parameter. Secondly, we take recall of fake class as our second parameter. We take this second parameter because we want to see how many of fake news are correctly predicted as fake news. Here, we give more emphasize on fake data over real or neutral data. In the above mentioned table, recall is for fake class. For English language, our independent model gives the better result in both parameters. For French, German, Italian and Spanish results of dependent and independent models are comparable in both parameters. For Hindi language, our independent model gives better results in case of both the parameters. It has also been observed that the Bengali language independent model gives better result in F1 score, but dependent model gives better result in recall value.

7 Conclusion

The present work deals with some of the modern day topics like fake news and COVID19. We first collect our data using various sources like twitter, newspaper and shared task. We analyse fake news in multilingual aspect and check how each model and language performs differently in each scenario. Though our data in Hindi and Bengali is very less but still we got some good results in some model. In future if we can get more data in Hindi and Bengali then we can build more concrete models upon these two languages. Here we also learn the basic architecture and concepts of BERT model which is one of the most popular pre-trained models of NLP. We build a language independent model using BERT multilingual which supports many languages. So with our collected data we just fine-tune BERT model with each language data. It produces some astonishing results. Though our data is less but it still gives very good results. In English, Hindi and Bengali language our language independent model outperformed most of the language dependent models. In case of European languages like German, French, Italian and Spanish both language dependent and language independent model performs very good and their results are comparable. One thing we should mention that in case of German language our language independent model predicted all real news correctly and only four fake news wrongly, which quite astonishing.

Acknowledgments

The present work is supported by the research project entitled " Claim Detection and Verification using Deep NLP: an Indian Perspective" funded by DRDO, Government of India..

References

George, J., Skariah, S., & Aleena Xavier, T. (2020). Role of Contextual Features in Fake News Detection: A Review. 2020 International Conference On Innovative Trends In Information Technology (ICITIIT). doi: 10.1109/icitiit49094.2020.9071524

P´erez-Rosas, V., Kleinberg, B., Lefevre, A., & Mihalcea, R. (2018). Automatic Detection of Fake News. In Proceedings of the 27th International Conference on Computational Linguistics (pp. 3391–3401). Santa Fe, New Mexico, USA: Association for Computational Linguistics.

Qi, P., Cao, J., Yang, T., Guo, J., & Li, J. (2019). Exploiting Multi-domain Visual Information for Fake News Detection. 2019 IEEE International Conference On Data Mining (ICDM). doi: 10.1109/icdm.2019.00062

Bedi, A., Pandey, N., & Khatri, S. (2019). A Framework to Identify and secure the Issues of Fake News and Rumours in Social Networking. 2019 2Nd International Conference On Power Energy, Environment And Intelligent Control (PEEIC). doi: 10.1109/peeic47157.2019.8976800

Dey, A., Rafi, R., Hasan Parash, S., Arko, S., & Chakrabarty, A. (2018). Fake News Pattern Recognition using Linguistic Analysis. 2018 Joint 7Th International Conference On Informatics, Electronics & Vision (ICIEV) And 2018 2Nd International Conference On Imaging, Vision & Pattern Recognition (Icivpr). doi: 10.1109/iciev.2018.8641018

Rajesh, K., Kumar, A., & Kadu, R. (2019). Fraudulent News Detection using Machine Learning Approaches. 2019 Global Conference For Advancement In Technology (GCAT). doi: 10.1109/gcat47503.2019.8978436

Uppal, A., Sachdeva, V., & Sharma, S. (2020). Fake news detection using discourse segment structure analysis. 2020 10Th International Conference On Cloud Computing, Data Science & Engineering (Confluence). doi: 10.1109/confluence47617.2020.9058106

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate.

Karimi, H., Roy, P., Saba-Sadiya, S., & Tang, J. (2018). Multi-Source Multi-Class Fake News Detection. In Proceedings of the 27th International Conference on Computational Linguistics (pp. 1546–1557). Santa Fe, New Mexico, USA: Association for Computational Linguistics.

Yichuan Li and Bohan Jiang and Kai Shu and Huan Liu (2020). MM-COVID: A Multilingual and Multimodal Data Repository for Combating COVID-19 Disinformation. CoRR, abs/2011.04088.

S.B. Majumder and D. Das (2020). Detecting Fake News Spreaders on Twitter Using Universal Sentence Encoder. CLEF

Fix, E. and Hodges, J.L. (1951) Discriminatory Analysis, Nonparametric Discrimination: Consistency Properties. Technical Report 4, USAF School of Aviation Medicine, Randolph Field.

Boser, B., Guyon, I., Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, Pittsburgh.

Breiman, L. (2001). Random Forests. Machine Learning, 45, 5-32.

Chen, T., Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785–794). New York, NY, USA: ACM.

Wikipedia Current Events Summarization using Particle Swarm Optimization

Santosh Kumar Mishra [1], **Darsh Kaushik** [2], **Sriparna Saha** [1], **and Pushpak Bhattacharyya** [3]
[1] Department of Computer Science & Engineering, Indian Institute of Technology Patna, India
[2] Department of Computer Science & Engineering, National Institute of Technology Silchar, India
[3] Department of Computer Science & Engineering, Indian Institute of Technology Bombay, India
{santosh_1821cs03, sriparna}@iitp.ac.in [1], darsh_ug@cse.nits.ac.in [2], pb@cse.iitb.ac.in [3]

Abstract

This paper proposes a method to summarize news events from multiple sources. We pose event summarization as a clustering-based optimization problem and solve it using particle swarm optimization. The proposed methodology uses the search capability of particle swarm optimization, detecting the number of clusters automatically. Experiments are conducted with the Wikipedia Current Events Portal dataset and evaluated using the well-known ROUGE-1, ROUGE-2, and ROUGE-L scores. The obtained results show the efficacy of the proposed methodology over the state-of-the-art methods. It attained improvements of 33.42%, 81.75%, and 57.58% in terms of ROUGE-1, ROUGE-2, and ROUGE-L, respectively.

1 Introduction

The continuously rising amount of text data makes analyzing and comprehending textual files tiresome as technology progresses in a fast-changing fashion. Capturing important information from large documents is a time-consuming and labor-intensive job from the reader's perspective. A large number of documents must be handled quickly, and a large amount of text data necessitates the use of text and document summarization algorithms. However, the focus has been on single-document summarization both for extractive and abstractive variants with comparatively little advancements in multi-document summarization.

Multi-document summarization techniques are becoming paramount in recent years. There are several real life applications of multi document summarization like : scientific summarization (Yasunaga et al., 2019) (Mishra et al., 2021d) (Mishra et al., 2020), news summarization (Fabbri et al., 2019), email thread summarization (Zhang et al., 2021), summarization of product reviews (Gerani et al., 2014), course feedback summarization (Luo et al., 2016), Wikipedia article generation (Liu et al., 2018), summarization of medical documents (Afantenos et al., 2005).

Deep learning has gained a huge amount of attention in recent years as a result of its success in computer vision (Krizhevsky et al., 2012), natural language processing (Devlin et al., 2014) (Mishra et al., 2020), and multi-modal applications (Wang et al., 2020) (Mishra et al., 2021b) (Mishra et al., 2021a). Researchers use deep learning to solve challenging problems because of its capacity to capture highly nonlinear data relationships. Deep learning-based models have recently been used in multi-document summarization (Zhang et al., 2021) (Fabbri et al., 2019) (Yasunaga et al., 2017), advancing the field of text summarization and allowing models to improve their performances. Attempts to utilise big deep learning models, which have considerably improved the state-of-the-art for different supervised natural language processing tasks, however, are hampered by a shortage of large datasets, making a comprehensive evaluation impossible. Even with larger datasets, compute resources and the corresponding training time might also pose a challenge in case of MDS (multi-document summarization) since several documents have to be processed. Moreover, for single and multi-document summarization, meta-heuristics algorithms have shown good results in previous studies (Mishra et al., 2021d) (Saini et al., 2019a) (Saini et al., 2019b) (Mishra et al., 2021c).

In this work, we have proposed a meta-heuristic optimization technique-based multi-document summarization methodology using Wikipedia Current Events Portal (WCEP) dataset introduced in Association for Computational Linguistics (ACL), 2020 (Ghalandari et al., 2020). Major contributions of this work are as follows:

- Employment of word mover's distance (WMD) (Kusner et al., 2015) to find the document center, it captures the semantic similarity. The proposed approach(s) utilize the word move distance (WMD) to capture the semantic similarity of sentences. It's worth noting that WMD doesn't require sentences to be represented as vectors. It employs word embeddings for various terms derived from a word2vec model trained on the Google news corpus, which comprises of 3 billion words and each word vector has 300 dimensions. If two phrases are similar, the WMD for each will be 0.

- We have used the particle swarm optimization-based clustering (PSO) (Kennedy and Eberhart, 1995) technique to cluster these news event sentences efficiently. It decides the number of clusters automatically within documents. This is the first effort to summarize Wikipedia's current event documents using PSO-based clustering to the best of our knowledge.

- To generate the summary, meaningful sentences from different clusters are selected using sentence scoring features like sentence's position, sentence length, similarity with paper's title, and similarity with the document center.

2 Related Work

To solve multi-document summarization, non-neural and neural network-based methods have been used in the literature.

Non-neural approaches have been widely used in the literature for multi-document summarization. In (Carbonell and Goldstein, 1998), authors have used query relevance and maximum marginal relevance to accomplish text summarization. They utilized the maximum marginal relevance to maintain anti-redundancy in generated summary. Authors of (Radev et al., 2004) proposed a clustering-based approach in which summary is generated using cluster centroid. Apart from this, they proposed evaluation techniques using subsumption and sentence utility for single and multi-document summarization. In (Erkan and Radev, 2004), an unsupervised graphical method, LexRank, has been proposed for text summarization. Here, the proposed method accomplishes sentence scoring using the graph-based method. LexRank finds the important sentences utilizing eigenvector centrality of graph representation denoting sentences. Authors of (Mihalcea and Tarau, 2004) have proposed the TextRank method for text summarization; this is based on page ranking methodology. In (Haghighi and Vanderwende, 2009), a generative probabilistic methodology to summarize multiple documents is proposed. Here, the authors have proposed a way of constructing a sequence of models using a frequency-based model. In (Radev and McKeown, 1998), authors have developed a method 'SUMMONS' that combines information from various news articles and converts it into a summary. In (Barzilay et al., 1999), multi-document summarization is accomplished by finding similar elements across texts from different documents. A graph-based summarization technique, namely 'Opinosis' introduced in (Ganesan et al., 2010), generates a precise abstractive summary from the redundant opinion. A word-level and sentence-level ranking based on various indicators of importance, keyword extraction, and phrase-level salience (Hong, 2005) (Cao et al., 2015), greedy heuristics on relation graphs and embedding (Yasunaga et al., 2017) has been used to solve the multi-document summarization.

Nowadays, supervised learning is used to solve extractive and abstractive summarization problems. But, limitation of supervised approaches is that it requires a huge amount of data for training. An attention with encoder-decoder based recurrent neural network is introduced in (Nallapati et al., 2016a). Here, authors have carried out abstractive summarization over DUC and CNN/Daily mail datasets. In (Cheng and Lapata, 2016), authors have proposed a data-driven approach with a deep neural network that incorporates the continuous sentence features. They developed an architecture consisting of a hierarchical document encoder and an attention-based extractor. A sentence ranking-based approach for single document summarization is explored in (Narayan et al., 2018). Here, authors have proposed a training methodology to optimize the ROUGE evaluation metric using reinforcement learning. A conditional recurrent neural network has been employed for abstractive summarization in (Chopra et al., 2016). Here, the convolutional attention-based encoder ensures the conditioning of the input sequence that helps the decoder to focus on relevant input words at each time step of the summary generation. In (Nallapati et al., 2016b), an extractive summarization of the document has been carried out using contrasting recurrent neural network-based architecture. The proposed method classifies the sentences in a sequential way that decides whether a sentence should be accepted or rejected to be included in the summary. Further, a sentence selector selects a single sentence at a time in random order to form the summary. A sequence to sequence architecture for abstractive summarization has been introduced in (See et al., 2017). Authors have proposed a pointer generator-based sequence to sequence model that can copy a word from the source text that helps in generating an accurate summary. In (Paulus et al., 2017), an intra-attention mechanism has been introduced that attends the input sequence and generates the output separately in a continuous manner. The authors have also proposed a new training methodology that utilizes reinforcement learning and supervised word prediction. Standard word prediction is coupled with RL's global sequence prediction training, resulting in more comprehensible summaries. Author of (Cohan et al., 2018) proposed a architecture to learn discourse structure of the documents. Apart from these, they also employed an attentive discourse-aware decoder that can summarize single and multiple documents. In (Celikyilmaz et al., 2018), abstractive summarization has been accomplished using deep communicating agents in the encoder-decoder model. Here, the deep communicating agents divide the long documents into smaller parts and assign them to different collaborative agents. The collaborative agents work as agents connected through a single decoder which trains end-to-end using reinforcement learning to generate a coherent and accurate summary. In (Gehrmann et al., 2018), authors have introduced a data-efficient content selector that finds the phrase in the input document that is important for the summary. This selector is employed as bottom-up attention to constraining the model to similar phrases.

The limitation of the supervised approach (deep learning model) is that it needs a huge amount of data for learning. We often don't have enough data to train a supervised model in many instances. Motivated by this, we present an unsupervised method for summarizing events in an extractive way from recent news, which we evaluate on the WCEP dataset (Ghalandari et al., 2020). It contains daily news events and their corresponding summaries. The proposed approach does not require massive data, and it has consistent performance irrespective of dataset size.

3 Proposed Methodology

This section has an overview of the proposed methodology. Fig 1 and Algorithm 1 illustrate the steps and pseudo-code, respectively. The notations used in this section are defined in Table 1.

The proposed methodology is based on a natural phenomenon; at the end of its execution, it generates a set of solutions. We get a set of optimal solutions at the end. Here, a solution is made up of a group of sentence clusters that have been optimized (particle).

Particle swarm optimization (Kennedy and Eberhart, 1995) is a famous nature-inspired method that was designed inspired by the social behavior of bird flocks. It's a population-based method of searching. The method maintains a population of particles. Every particle in this diagram represents a viable optimization solution. A swarm comprises numerous alternative solutions to an optimization issue known as particles in the PSO framework. The PSO algorithm's goal, in this case, is to find the optimal particle position that produces the best fitness value in terms of the objective function. We used a PSO-based clustering approach with K-means clustering to seed the original swarm. It entails the following procedures:

1. Particle representation: Each particle chooses K different sentence vectors as initial cluster centroid vectors in the first step.

2. - Points are assigned to various clusters as follows: Each phrase vector is allocated to the centroid vector that is closest to it, and then the fitness value is calculated using Equation 5.
   - Updation of position and velocity: In order to create the new solution, the particle's velocity and position are changed using Equations 1 and 2.

3. Step 2 should be repeated until the termination condition is met:
   - The total number of iterations has been achieved.
   - There is a little change in the centroid vector.

Each particle in N_d dimensional space represents a position, and it moves throughout multi-dimensional search space, changing its location in reference to both:

- Particle's best position found.
- Best position in the neighborhood of that particle.

The following information is maintained by every particle:

- y_i The particle's personal best position.
- x_i: The particle's current position.
- v_i The particle's current velocity.

Using the notations above, the particle's position is modified according to

$$\begin{aligned} v_{i,k}(t+1) = wv_{i,k}(t) + c_1 r_{1,k}(t)(y_{i,k}(t) \\ -x_{i,k}(t)) + c_2 r_{2,k}(t)(\hat{y_k}(t) - x_{i,k}(t)) \end{aligned} \tag{1}$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{2}$$

Here, c_1 and c_2 is denotes the acceleration constant, inertia weight is w, $r_{1,j}(t)$, $r_{2,j}(t)$ denote the random number between 0 and 1 where $k = 1, ..., N_d$. Velocity is computed using three components: (1) component denoting function of particle's distance from the personal best position, (2) fraction of the previous velocity, (3) social component representing the distance between particle and best particle.

The particle's personal best position is measured as follows:

$$\begin{aligned} y_i(t+1) = y_i(t) \;\; if \;\; f(x_i(t+1)) \geq f(y_i(t)) \\ x_i(t+1) \;\; if \;\; f(x_i(t+1)) < f(y_i(t)) \end{aligned} \tag{3}$$

Equation 1 denotes the global best version of PSO, where at end global best solution is taken into consideration, where i^{th} particle's neighborhood has best particle $\hat{y_k}$.

A single particle describes the N_c centroid vectors in the sense of clustering. Here, every particle x_i is formed, as follows:

$$x_i = \{m_{i1}, m_{i2}, ...m_{ij}, ...m_{iN_c}\} \tag{4}$$

Here, m_{ij} corresponds to the centroid vector of the j^{th} cluster of the i^{th} particle of the C_{ij} cluster; therefore, for the existing data vectors, a swarm describes a set of candidate clusters. As a quantization error, the fitness of the particle is calculated as follows:

$$E = \frac{\sum_{j=1}^{N_c} [\sum_{\forall Z_p \in C_{ij}} d(z_p, m_j)/|C_{ij}|]}{N_c} \tag{5}$$

PSO begins with a swarm, which is a collection of possible solutions (particles). The particle X_i is made up of solutions $\{m_{i1}, m_{i2}, ...m_{ij}, ...m_{iN_c}\}$ with varying numbers of clusters. Our assumption is the solution, m^{ij}, would represent the centers of the sentence clusters. However, this is a challenging task to identify the number of clusters in a document automatically. Because of this complexity, each solution has a different number of clusters, ranging from $1 \leq K \leq M$. M signifies the number of sentences to be clustered, using K number of clusters.

K-mean algorithm, with the current number of cluster centers, is invoked for each solution. After each iteration of the K-means algorithm, cluster centroids/centers are modified, and this step is repeated until the centroids are converged. particles change the velocity and position to obtain the best fitness values. In the end, it automatically decides the number of clusters as the algorithm terminates.

3.1 Summary Generation

The summary generation procedure is as follows:

- **Document's center identification:** The sentence with the lowest average WMD distance is considered as the document center with respect to all other sentences. The M number of sentences are taken into account to determine the average WMD for a sentence.

$$t = argmin_i \sum_{i=1}^{M} \sum_{j=1, i \neq j}^{M} \frac{wmd(s_i, s_j)}{A} \tag{6}$$

  Where, representative sentence or document centre is t, M denotes the number of news sentences , s_i, s_j denote document's i^{th} and j^{th} sentence, respectively, A represents the number of sentence pairs.

- **Cluster's ranking in i^{th} particle:** The Cosine similarity computed between the cluster centers and the document center are used to rank clusters inside a particle in decreasing order. In other words, cluster with the shortest distance to the document center will be given higher priority(higher rank) than others.

In order to generate the summary, sentences belonging to different clusters (high to low) must be extracted as per their ranks.

Sentence scores are therefore assessed in each cluster on the basis of four features. Those are the similarity of sentence to the paper's title, length of the sentence, the position of the sentence, sentences close to the document center, Descriptions of these features are given below:

- Similarity with the paper's title (F1): The sentences which are semantically close to the document's title have given high scores (Saini et al., 2019a). Firstly, these sentences are considered summary generation. This is defined as follows:

$$F1 = wmd(s_i^k, title) \tag{7}$$

  where, s_i^k represents i^{th} sentence of the k^{th} cluster, document's title is represented by $title$ and $dist_{wmd}$ is WMD between sentence and document's title.

- Position of the sentence (F2:) Essential sentences can be found at the start of most paragraphs/documents. These sentences can be helpful to generate a good quality summary (Saini et al., 2019a).

$$F2 = \frac{1}{\sqrt{r}} \tag{8}$$

- Length of sentence (F3:) The length of sentence has been used as a selecting criteria. Here the sentence which are longer in the length given higher priority over others (Saini et al., 2019b) (Mishra et al., 2021d).

- Sentences close to the document center in terms of WMD (F4): Sentences in each cluster identical to the document center in terms of WMD have been included first in summary (Saini et al., 2019b) (Saini et al., 2019a).

Symbol	Meaning
N_c	Number of cluster centroids
x_i	Particle's current position
t	Time steps
v_i	Particle's current velocity
N_c	Cluster centroid vector
y_i	Particle's best position
N_d	Input dimension
$v_{i,k}(t+1)$	Updated velocity in k dimension
w	weight of inertia
r	Random number between 0 and 1
$\hat{y}$	Best particle
m	Centroid Vector
c_1, c_2	Acceleration constant
WMD	Word mover's distance

Table 1: List of abbreviations

Algorithm 1 WCEP_EventSumm-PSO

```
1: **Input:** News event from Wikipedia Current Event Portal
2: **Output:** Summary of the news events
3: Initialize each particle with $N_c$ randomly selected centroids.
4: for $i \leftarrow 1$ to $t_{max}$ do
5:     for each particle $i$ do
6:         for each data vector $z_p$ do
7:             Find the Euclidean distances $d(z_p, m_{ij})$ to all cluster centroids $C_{ij}$
8:             $z_p$ assigned to cluster $C_{ij}$ such that $d(z_p, m_{ij}) = min_{\forall c=1,\dots\dots\dots N_c}\{d(z_p, m_{ic})\}$
9:             Calculate the fitness using Equation 5
10:    Global and local best positions are being updated.
11:    Update the centroids of the clusters using Equations 1 and 2.
12:    Summary generation corresponding to Global best solutions as discussed in section 3.1
```

Methods	ROUGE-1	ROUGE-2	ROUGE-L
Similarity with the title (F1)	0.459	0.239	0.395
Position of the sentence (F2)	0.471	0.249	0.405
Length of the sentence (F3)	0.425	0.203	0.355
Similarity with the document center (F4)	0.442	0.221	0.376

Table 2: Score obtained with different features

4 Experimental Setup

This section has a detailed discussion on dataset and evaluation metrics used.

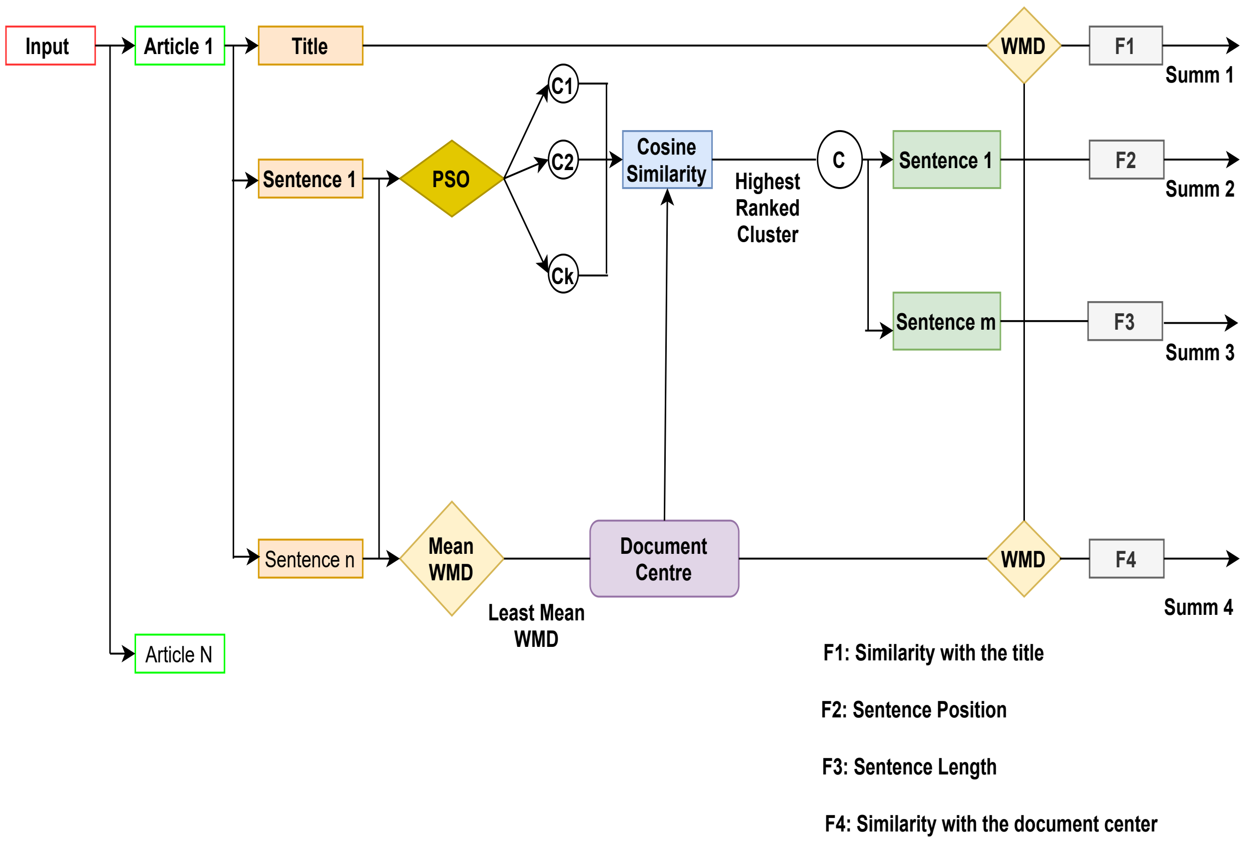


Figure 1: The process flow chart of the proposed method.

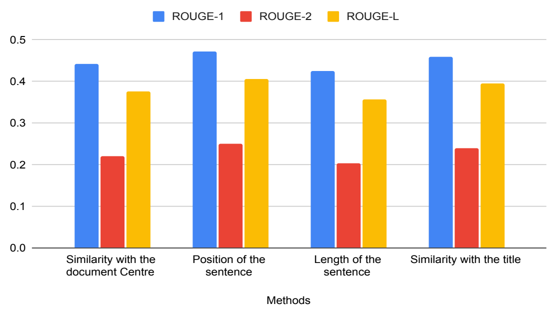


Figure 2: The bar graph of obtained score with the different features

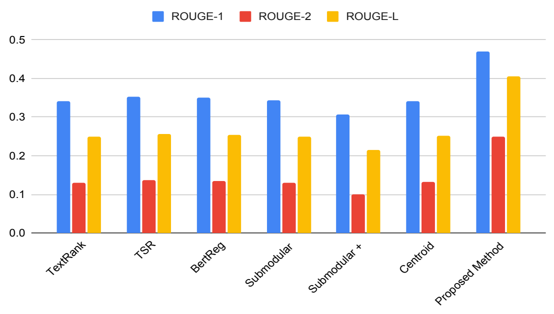


Figure 3: The bar graph showing comparison with the state-of-the-art methods

4.1 Dataset

We use WCEP (Ghalandari et al., 2020) dataset for the experimentation. The dataset contains 10,200 items from recent news events, as well as their summaries. Train set, validation set and test set consist of 8158, 1020 and 1022 respectively.

4.2 Evaluation metrics

The proposed approach is evaluated using the popular evaluation metrics ROUGE scores (Lin, 2004) used for text and document summarization. This score computes the overlapping n-grams between the generated summary and the ground truth summary. F1-score, precision, and recall are commonly utilized in the literature to do quantitative analysis. The Rouge-F1 scores are shown in the Tables 2 and Table 3 .

5 Result and Discussions

This section has a detailed discussion on results obtained and their analysis. We have shown the obtained score in Table 2 and comparison with the

Methods	ROUGE-1	ROUGE-2	ROUGE-L
TextRank	0.341	0.131	0.25
TSR	0.353	0.137	0.257
BertReg	0.35	0.135	0.255
Submodular	0.344	0.131	0.25
Submodular + Abs	0.306	0.101	0.214
Centroid	0.341	0.133	0.251
Proposed Method	**0.471**	**0.249**	**0.405**

Table 3: Comparison of the proposed method with the state-of-the-art methods

state-of-the-art methods in Table 3.

5.1 State-of-the-art comparative baselines

We have accomplished the comparison with the following state-of-the-art methods:

- *TextRank:* This is an unsupervised method of text summarization (Mihalcea and Tarau, 2004). It is based on a graph-based ranking model that perceives the most important sentence and the keyword for the summary.
- *Centroid:* This methodology generates the summary utilizing the cluster centroid generated by a topic detection algorithm (Radev et al., 2004).
- *TSR:* This approach is based on sentence ranking based on statistical feature an average of the word embedding vectors (Ren et al., 2016).
- *BERTREG:* This is similar to TSR methodology, but it uses the sentence embedding produced by pre-trained BERT (Devlin et al., 2019).
- *SUBMODULAR:* This method is based on the submodular function that integrates coverage and non-redundancy to find the important sentence within the document to form the summary (Chali et al., 2017).
- *SUBMODULAR + Abs:* Abstractive based approach sentence compression and metging is incorporated in SUBMODULAR approach (Chali et al., 2017).

5.2 Analysis of the Results:

We have shown the obtained score with different features in Table 2 and comparison with state-of-the-art methods in 3. It can be concluded from Table 3 that the proposed methodology outperforms the state-of-the-method. It can be seen from Table 3 that TSR (Ren et al., 2016) has the highest score among all methods. The bar graph of scores obtained with different features and comparison with the state-of-the-art is shown in Fig 2 and Fig 3 respectively. If, we compare with the TSR method, the proposed method has the improvement of 33.42%, 81.75%, and 57.58% considering ROUGE-1, ROUGE-2, and ROUGE-L, respectively.

6 Conclusion and Future Works

This paper presents a method of Wikipedia current event summarization using a particle swarm optimization-based clustering methodology. We utilized the search capability of particle swarm optimization as an underlying optimization strategy, an evolutionary algorithm. The proposed method detects the number of clusters automatically. The different feature has been employed for sentence scoring within-cluster and to form the final summary. The efficacy of the proposed method has been tested on the WCEP dataset. The obtained results show the effectiveness of the proposed method over state-of-the-art methods. Compared to the best method among the state-of-the-art, the proposed method has the improvement of 33.42%, 81.75%, and 57.58% in terms of ROUGE-1, ROUGE-2, and ROUGE-L, respectively.

In the future, this work can be extended using the ensembling of the clustering technique. Apart from that, more sophisticated feature word mover's distance, BERT similarity, and textual entailment can be utilized for a summary generation.

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

Stergos Afantenos, Vangelis Karkaletsis, and Panagiotis Stamatopoulos. 2005. Summarization from medical documents: a survey. *Artificial intelligence in medicine*, 33(2):157–177.

Regina Barzilay, Kathleen McKeown, and Michael Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, pages 550–557.

Ziqiang Cao, Furu Wei, Li Dong, Sujian Li, and Ming Zhou. 2015. Ranking with recursive neural networks and its application to multi-document summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. *arXiv preprint arXiv:1803.10357*.

Yllias Chali, Moin Tanvee, and Mir Tafseer Nayeem. 2017. Towards abstractive multi-document summarization using submodular function-based framework, sentence compression and merging. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 418–424.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252*.

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.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. *arXiv preprint arXiv:1804.05685*.

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.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *proceedings of the 52nd annual meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1370–1380.

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.

Alexander R Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir R Radev. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. *arXiv preprint arXiv:1906.01749*.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: A graph based approach to abstractive summarization of highly redundant opinions.

Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. 2018. Bottom-up abstractive summarization. *arXiv preprint arXiv:1808.10792*.

Shima Gerani, Yashar Mehdad, Giuseppe Carenini, Raymond Ng, and Bita Nejat. 2014. Abstractive summarization of product reviews using discourse structure. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1602–1613.

Demian Gholipour Ghalandari, Chris Hokamp, Nghia The Pham, John Glover, and Georgiana Ifrim. 2020. A large-scale multi-document summarization dataset from the wikipedia current events portal. *arXiv preprint arXiv:2005.10070*.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of human language technologies: The 2009 annual conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370.

Gumwon Hong. 2005. Relation extraction using support vector machine. In *Second International Joint Conference on Natural Language Processing: Full Papers*.

James Kennedy and Russell Eberhart. 1995. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.

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. PMLR.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*.

Wencan Luo, Fei Liu, Zitao Liu, and Diane Litman. 2016. Automatic summarization of student course feedback. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 80–85.

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.

Santosh Kumar Mishra, Rijul Dhir, Sriparna Saha, and Pushpak Bhattacharyya. 2021a. A hindi image caption generation framework using deep learning. *Transactions on Asian and Low-Resource Language Information Processing*, 20(2):1–19.

Santosh Kumar Mishra, Rijul Dhir, Sriparna Saha, Pushpak Bhattacharyya, and Amit Kumar Singh. 2021b. Image captioning in hindi language using transformer networks. *Computers & Electrical Engineering*, 92:107114.

Santosh Kumar Mishra, Harshavardhan Kundarapu, Naveen Saini, Sriparna Saha, and Pushpak Bhattacharyya. 2020. Iitp-ai-nlp-ml@ cl-scisumm 2020, cl-laysumm 2020, longsumm 2020. In *Proceedings of the First Workshop on Scholarly Document Processing*, pages 270–276.

Santosh Kumar Mishra, Naveen Saini, Sriparna Saha, and Pushpak Bhattacharyya. 2021c. Let's summarize scientific documents! a clustering-based approach via citation context. In *International Conference on Applications of Natural Language to Information Systems*, pages 330–339. Springer.

Santosh Kumar Mishra, Naveen Saini, Sriparna Saha, and Pushpak Bhattacharyya. 2021d. Scientific document summarization in multi-objective clustering framework. *Applied Intelligence*, pages 1–24.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016a. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Ramesh Nallapati, Bowen Zhou, and Mingbo Ma. 2016b. Classify or select: Neural architectures for extractive document summarization. *arXiv preprint arXiv:1611.04244*.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. *arXiv preprint arXiv:1802.08636*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Dragomir Radev and Kathleen McKeown. 1998. Generating natural language summaries from multiple online sources. *Computational Linguistics*, 24(3):469–500.

Dragomir R Radev, Hongyan Jing, Małgorzata Styś, and Daniel Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing & Management*, 40(6):919–938.

Pengjie Ren, Furu Wei, Zhumin Chen, Jun Ma, and Ming Zhou. 2016. A redundancy-aware sentence regression framework for extractive summarization. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 33–43.

Naveen Saini, Sriparna Saha, Anubhav Jangra, and Pushpak Bhattacharyya. 2019a. 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.

Naveen Saini, Sriparna Saha, Anurag Kumar, and Pushpak Bhattacharyya. 2019b. Multi-document summarization using adaptive composite differential evolution. In *International Conference on Neural Information Processing*, pages 670–678. Springer.

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

Hu Wang, Qi Wu, and Chunhua Shen. 2020. Soft expert reward learning for vision-and-language navigation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*, pages 126–141. Springer.

Michihiro Yasunaga, Jungo Kasai, Rui Zhang, Alexander R Fabbri, Irene Li, Dan Friedman, and Dragomir R Radev. 2019. Scisummnet: A large annotated corpus and content-impact models for scientific paper summarization with citation networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7386–7393.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. *arXiv preprint arXiv:1706.06681*.

Shiyue Zhang, Asli Celikyilmaz, Jianfeng Gao, and Mohit Bansal. 2021. Emailsum: Abstractive email thread summarization. *arXiv preprint arXiv:2107.14691*.

Automated Evidence Collection for Fake News Detection

Mrinal Rawat
UpGrad Education Pvt. Ltd., Mumbai, India.
Liverpool John Moores University,
Liverpool, United Kingdom.
rawatmrinal06@gmail.com

Diptesh Kanojia
Centre for Translation Studies,
University of Surrey,
Surrey, United Kingdom.
d.kanojia@surrey.ac.uk

Abstract

Fake news, misinformation, and unverifiable facts on social media platforms propagate disharmony and affect society, especially when dealing with an epidemic like COVID-19. The task of Fake News Detection aims to tackle the effects of such misinformation by classifying news items as fake or real. In this paper, we propose a novel approach that improves over the current automatic fake news detection approaches by automatically gathering evidence for each claim. Our approach extracts supporting evidence from the web articles and then selects appropriate text to be treated as evidence sets. We use a pre-trained summarizer on these evidence sets and then use the extracted summary as supporting evidence to aid the classification task. Our experiments, using both machine learning and deep learning-based methods, help perform an extensive evaluation of our approach. The results show that our approach outperforms the state-of-the-art methods in fake news detection to achieve an F1-score of 99.25 over the dataset provided for the CONSTRAINT-2021 Shared Task. We also release the augmented dataset, our code and models [1] for any further research.

1 Introduction

The ability to consume readily available information from the internet is alarming for both individuals and organizations. The quality of content on social media platforms has been significantly affected due to the spread of fake news, misinformation and unverifiable facts. The current tally of internet users stands at 4.66 billion[2] (Kemp, 2015); and many of these users generate, post and consume content without any regulation, in a large number of countries[3]. Due to the unrestricted nature of online platforms, there is a significant increase in the amount of misinformation on social media (Allen et al., 2020), especially in developing nations (Badrinathan, 2020; Wasserman and Madrid-Morales, 2019). Studies show that events such as the presidential election of the United States in 2016 were affected due to *moderated fake news* campaigns (Tavernise, 2016). Shu et al.(2017) (Shu et al., 2017) propose that *fake news is intentionally written, verifiably false, and is created in a way that makes it look authentic*. Manual efforts by other online platforms such as Poynter[4], FactCheck[5], AltNews[6] *etc.* to detect fake news, requires a lot of human effort and can prove to be cumbersome. Such manual efforts can be time-consuming, challenging, and at times, can also be ineffective as fake news can spread faster than verified claims over social media platforms.

Automatic Fake News Detection is a task that aims to mitigate the problem of misinformation with the help of evidence supported by various sources. Most of the approaches in this recently devised task aim to use the classical machine learning-based methods or the recent deep learning-based methods to help classify news items as fake or as real. Initially proposed methods for the task applied machine learning-based techniques but cited insufficient data as a major concern (Vlachos and Riedel, 2014). Recent deep learning and ensemble approaches (Malon, 2018; Roy et al., 2018) were proposed on the FEVER (Thorne et al., 2018a) and LIAR (Wang, 2017) datasets, and have been shown to perform very well. Studies have proposed a combination of evidence detection with textual entailment

[1] https://github.com/rawat-mrinal06/fake_news
[2] Internet Live Stats (as on 15-07-2021)
[3] Internet Censorship in Countries
[4] Poynter: Online
[5] FactCheck: Online
[6] AltNews: Online

concerning the claim (Vijjali et al., 2020). FEVER Shared Tasks (Thorne et al., 2018b, 2019) have helped the automatic fact verification task gather attention towards the problem and helped generate approaches to mitigate the issues with previously proposed solutions. This study shows that our novel approach improvises over state-of-the-art approaches and helps detect fake news related to COVID-19. Our approach performs web-search for evidence collection and uses BERT-score similarity to match the unverified claim with the top-k searches. Further, we propose the use of summarization to mitigate problems with the evidence collection. We summarize the top-n selected lines from these articles and use them as evidence to support or reject the news item claim. Our experiments perform an extensive evaluation of the approach over the datasets released as a part of the CONSTRAINT-2021 Shared Task (Patwa et al., 2020) shared task.

Our summarized contributions with this paper are:

- We propose a novel approach to help automate the evidence collection for any fake news detection dataset.
- Additionally, we incorporate a summarization component that helps outperform the state-of-the-art approaches for automatic fact verification on the CONSTRAINT-2021 dataset.

2 Related Work

Automatic detection and classification of fake news, especially in epidemic situations like COVID-19, is a significant issue for society. Most of the recent works have identified that fake news is written intentionally and factually false (Shu et al., 2017). Several datasets have been released for the AI community in the field of fake news detection, such as LIAR (Wang, 2017), Fake News Challenge-1, and FEVER (Thorne et al., 2018a). Some recent techniques extract the evidence from Wikipedia to classify a claim as `SUPPORTED`, `REFUTED` or `NOTENOUGHINFO` (Thorne et al., 2018a). They formulate the problem as a three-step process (i) first the top-k documents are identified based on the TF-IDF based approaches (ii) then top-k sentences are identified from the documents, and (iii) finally the textual entailment based approaches (Parikh et al., 2016) are used to classify the claim. Team Papelo (Malon, 2018) used the Transformer-based approach for the textual entailment and selected the evidence-based on tf-idf and entities present in the title. Hanselowski et al. (2018) selects the documents and sentence using the entity mentions and recognizes textual entailment using Enhanced Sequential Inference Model (ESIM) (Chen et al., 2017). Despite the several attempts, fake news detection is a challenging problem and countering fake news is a typical issue that requires continuous studies. Recently, some researchers released the datasets related to COVID-19 fake news detection. Shahi and Nandini (2020) (Shahi and Nandini, 2020) proposed the first multi-lingual cross-domain dataset for COVID-19 that consists of 5182 fact-checked news articles from Jan-2020 to May-2020. They collected data from 92 different websites and manually classified them into 23 classes. Kar et al. (Kar et al., 2020) also released a multi-indic-lingual dataset besides English to detect the fake news in social media tweets. They obtained 480 tweets in Bengali and 460 tweets in Hindi. In addition to tweets, they also included several features related to tweets such as retweet count, favourite count, total URL in description, URL, friend list, followers, *etc.* Recently, a very relevant dataset was released by Patwa et al. (Patwa et al., 2020) which consists of 10,700 tweets or claim collected from various sources such as Twitter, PolitiFact, Snopes, Boomlive. They experimented with various machine learning techniques like Decision Trees, Logistic Regression, SVM, Gradient Boosting DT and achieved the F1-score of 93.32. Most of the previous work on COVID-19 dataset proposed an ensemble approach of various models such as BERT, RoBERTa, XLNet, *etc.* (Shifath et al., 2021; Raha et al., 2021; Shushkevich and Cardiff, 2021). Chen et al. (Chen et al., 2021) trained the model with additional words such as covid-19, coronavirus, pandemic, indiafightscorona since the BERT tokenizer will split these words into separate tokens. Some works leveraged the fine-tuned models like COVID-Twitter-BERT (CT-BERT) (Müller et al., 2020) and demonstrated a boost in performance (Li et al., 2021; Glazkova et al., 2020; Wani et al., 2021). The fake news detection methods described above mainly uses the claim for the classification. Our method focuses on extracting and summarizing the evidence from the external source and uses it to classify the claim on the COVID-19 fake news detection dataset (Li et al., 2021; Patwa et al., 2020).

Split	**Real**	**Fake**	**Total**
Training	3360	3060	6420
Validation	1120	1020	2140
Test	1120	1020	2140
Total	5600	5100	10700

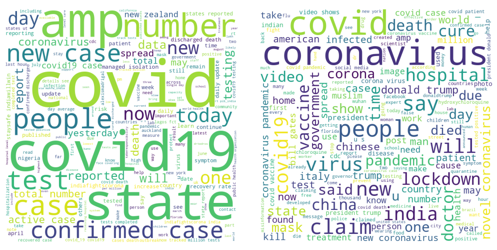
Table 1: Dataset Statistics

3 Dataset

For our work, we use the pre-released COVID-19 fake news dataset as a part of the CONSTRAINT-2021 shared task (Patwa et al., 2020). This gold-standard manually annotated dataset comprises social media posts and articles which are related to COVID-19. Each post or tweet contains content in the English language and is classified in either of the two categories- **(1) Real:** where tweets or articles which are factually correct and verified from authentic sources, for example, "Wearing mask can protect you from the virus. (Twitter)"; or **(2) Fake:** where tweets or posts related to COVID-19 which are factually incorrect and verified as false, for example, "If you take Crocin thrice a day you are safe. (Facebook)".

The authors collect fake news from two different sources- social media platforms and public fact-checking platforms. The social media posts include text from Facebook posts, Instagram posts, and Twitter posts, whereas the fact-checking websites such as PolitiFact, Snopes, and Boomlive are used to collect fact-checked news items. To further collect real news, they sample tweets from official government channels, news channels, and medical institutes. Overall, a total of 14 such sources were used to prepare this dataset.

The dataset comprises 10700 manually annotated samples and is split into (60%) train, (20%) validation and (20%) test sets. We provide the exact numbers for each split/class in Table 1 for clarity. The dataset is class-balanced as it contains 52.3% samples of real posts and 47.7% samples of fake posts. As an analysis on it, we obtained a word-cloud illustration for both real and fake samples and observed a high lexical overlap between both the classes, where words like '*coronavirus*', '*covid19*', '*people*', '*cases*', '*number*', '*test*', *etc.* are repeatedly used in both the sets. We do not show the word cloud due to space constraints. We create and present a wordcloud for the dataset in Figure 1.



(a) Worcloud of Real Posts (b) Worcloud of Fake Posts

Figure 1: Wordcloud of real/fake posts in our dataset.

4 Our Approach

In this section, we provide details of our novel approach to augment the dataset with evidence from web search and the use of this evidence to complement the task of fact verification. The algorithm for our approach can be seen in Algorithm 1. As discussed above, we collect this evidence and prune to top-k related news items based on semantic similarity via BERTScore (Devlin et al., 2019). We also select top-n lines from each article for further building an evidence repository, as detailed below in further subsections.

4.1 Evidence Collection

In the original dataset of COVID-19, evidence is not released along with the claim. We hypothesize that evidence is equally relevant to classify the claim as proposed by Thorne et al. (2018a). As per our approach, given a claim or post text, we first select K relevant articles using a BERT-based sentence similarity score as detailed here; and can be seen as an architecture component in Figure 2.

4.1.1 Article Retrieval

For each claim c, we search the claim as a query using a publicly available search API. The response returned by this API consists of `(heading, text)` pairs. We use the spacy (Honnibal et al., 2020) library to get the similarity score of response text with respect to the input claim. Based on this similarity score, we select top K results that have the similarity score greater than 0.7[7]. While selecting documents, we prune for webpages in other languages and pages which are direct links to PDF or other such non-text files. As an immediate next step, we scrape the selected web pages to obtain the matching N sentences concerning the claim as detailed here.

[7]Selected with empirical evaluation and manual analysis after trying 0.5, 0.6, 0.7, 0.8 using the 'en_nli_roberta_base' model for document similarity

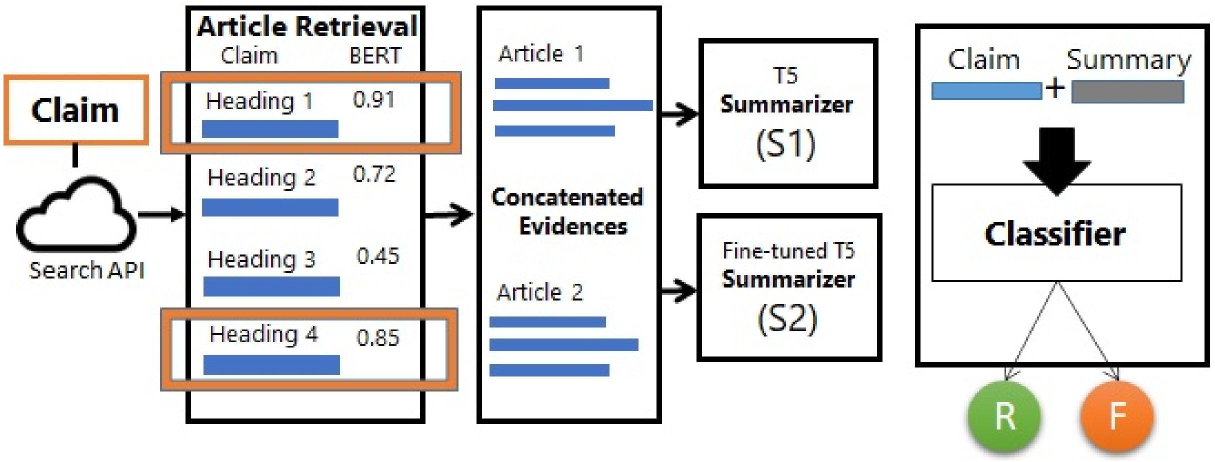


Figure 2: Full architecture of our proposed approach.

Algorithm 1: Algorithm to collect the evidence from the input claim

```
Input: Claim $c$, Blocked URLs $u$
Output: Evidence $e$
1  Function ArticleRet ($c$, $k = 3$):
2      $results \leftarrow GoogleSearch$(c);
       $filtered\_results \leftarrow \emptyset$; foreach
       $r_i \in results$ do
3          if $r_i \notin u$ then
4              $s_i \leftarrow$ Similarity($c$, $r_i$);
                // Document Similarity
                using spacy library
5              if $s_i > 0.7$ then
6                  $filtered\_results \leftarrow$
                    $(r_i, s_i)$
7      $filtered\_results \leftarrow$
        $Sort(filtered\_results)$[:k];
        return $filtered\_results$;
8  $articles \leftarrow$ ArticleRet ($c$);
9  $e \leftarrow \emptyset$;                    // Evidences
10 foreach $a_i \in articles$ do
11     $d \leftarrow WebsiteData(a_i.url)$;
        $sents \leftarrow d['<h>'] + d['<p>']$;
        // Extract <p> and <h> tags
        from html
12     foreach $s_i \in sents$ do
13         $sim \leftarrow Similarity(c, s_i)$; if
            $sim > 0.5$ then
14             $e \leftarrow (s_i, sim)$;
15     $e \leftarrow Sort$(e)[:3];
16 return $e$;
```

4.1.2 Sentence(s) Retrieval

In the previous step, we extract the relevant article URLs $U = (u_1, u_2, u_3)$. We employ a similar method to find the sentences within each article. For every url u, we first scrape the webpage and extract the text from $< h >$ and $< p >$ tags. Further, we use the same similarity score to select the top N sentences with respect to the claim. We obtain a similarity threshold of 0.5 after performing a similar empirical evaluation as mentioned in the footnote. Eventually, we concatenate the selected sentences from these articles, which act as our evidence for the claim. We would like to note that increasing the threshold significantly higher returned an empty set in some case and hence we choose a relatively lower threshold (0.5).

An example of evidence collected via our approach is shown in Table 2 where the column titled "Evidence" shows the output after these steps.

4.2 Dataset Preprocessing

To map claims with evidence, we pre-process both the dataset and the evidence collected from external sources. Following are the details of the preprocessing steps: **(1) URL Mapping:** We observe that some posts contain URLs in a masked form, *e.g.*, https://t.co/z5kk XpqkYb. Our approach extracts these URLs using a regular expression-based match and maps them to the original URL using the python '*requests*' library. Any additional information from the URL is removed, and only appropriate URLs remain in the text. For example, https://t.co/z5kkXpqkYb → https://www.cdc.gov/; **(2) Special symbols:** We removed extra whitespaces, special symbols and brackets like ;, (,), {,};

Claim	Evidence	Summarization-1 (S1)	Summarization-2 (S2)
There is no evidence that children have died because of a COVID-19 vaccine. No vaccine currently in development has been approved for widespread public use. https://t.co/9ecvMR8SAf	Currently there is no coronavirus vaccine that has been approved for the American public. And there is no evidence that children have died because they received one of the COVID-19 vaccines being developed. PolitiFact found no evidence that anyone has died from complications related to a trial COVID-19 vaccination. There is no evidence that children have died because of a COVID-19 vaccine.	There is no evidence that children have died because they received a COVID-19 vaccine. No evidence that anyone has died from complications related to a trial COVID-19.	There is no evidence that children have died because they received one of the COVID-19 vaccines being developed. PolitiFact found no evidence that anyone has died from complications related to a trial COVID-19 vaccination.

Table 2: Illustrative example of our approach pipeline shown as Claim → Evidence → Summarization-1 → Summarization-2, where Summarization-2 is obtained after fine-tuning T5 language model, and used as **evidence input** for classification

(3) Hashtags, Emojis and Mentions: Additionally, we remove hashtags and replace it with the token "HASHTAG:".

For example, #COVID-19 becomes HASHTAG:COVID-19. Similarly, we also replace mentions "@" with "MENTION:" token. At the end, we convert emojis to their text form using the 'demoji' library[8]; **(4) Lowercasing:** Eventually, we lowercase the claim and the evidence text to obtain the input data used for the next Summarization step.

4.3 Summarization of Evidence

Our pre-processed evidence for claims in many cases were multiple paragraphs resulting in performance degradation. Therefore, we propose the addition of a summarization component to our pipeline which utilizes state-of-the-art Text-to-Text Transfer Transformer (T5) language model (Raffel et al., 2019) for the inherent summarization task[9]. Due to the nature of the usual summarization task input, a large body of text (full documents), we believe that our comparatively short paragraphs would be better summarized. This language model is fine-tuned for the task of summarization helps us obtain a summarized text for each piece of evidence resulting in what we call Summarization-1 or S1. An output obtained is shown in Table 2.

4.3.1 Fine-tuning T5 on FEVER Dataset

As an additional experimental step, we further fine-tune the Text-to-Text Transfer Transformer (T5) model using the original FEVER dataset (Thorne et al., 2018a). The original T5 summarization model is trained on the CNN/Daily Mail (Hermann et al., 2015) data where the input is the news article text, and the objective is to highlight summarized text as the output. The T5 is an encoder-decoder model pre-trained on a multi-task mixture of unsupervised and supervised tasks and for which each task is converted into a text-to-text format. This allows for the use of the same model, loss function, hyperparameters, *etc.* across our diverse set of tasks. T5 works well on a variety of tasks out-of-the-box by prepending a different prefix to the input corresponding to each task, e.g., for the task of translation→ *translate English to German: <English Sentence>*, for the task of summarization→ *summarize: <English Text>*.

For our experiments, the aim is to summarize the pre-processed evidence while including the claim. Thus, we hypothesize that fine-tuning on an auxiliary dataset will improve the quality of the generated summary. For fine-tuning, we use the same hyperparameters as described in their paper to generate another model. We perform another iteration of the summarization step using this fine-tuned model to generate a parallel set of evidence and label the output as Summarization-2 or S2 as shown in Table 2. Further, we provide the details of **as the classification task, which uses either S1 or S2 as evidence input to classify** the claims as real or fake (Figure 2).

5 Experiment Setup

In this section, we discuss the experiment setup in detail. We perform the task of fake news detection as a binary classification task in a supervised setting. We choose to perform our experiments with both conventional machine learning- and deep learning- based classifiers. From the machine learning-based approaches, we choose Logistic Regression (LR) and Support Vector Machines (SVM) with the GridSearch implementation for best results over multiple hyperparameters (values of c, different kernels, *etc.*) We also utilize LSTMs with various contextual language models from the deep learning methods. From the deep learning-based approaches, we use a simple LSTM

[8]GitHub: Demoji

[9]This language model can perform the summarization task with the help of a prefix "summarize" to the input text provided.

	Previous Approaches		**Our Approach w/ various Classification methods**								
	Chen et al. (2021)	Li et al. (2021)	Logistic Regression			SVM			LSTM		
			-	S1	S2	-	S1	S2	-	S1	S2
P	0.9902	0.986	0.9531	0.9565	0.9701	0.9641	0.9671	0.9764	0.9589	0.9598	0.9612
R	0.9901	0.985	0.9531	0.9564	0.9700	0.9639	0.9668	0.9761	0.9584	0.9596	0.9612
F	0.9901	0.985	0.9531	0.9565	0.9700	0.9639	0.9668	0.9761	0.9584	0.9596	0.9612

Table 3: Results obtained after the fake news classification task where the values for previous approaches are from the latest shared task results and the results for each iteration of our approach are shown [P (Precision), R (Recall), and F (F-Score)]. (-) $\rightarrow$ No Evidence, S1 $\rightarrow$ Summarization-1 as Evidence, S2 $\rightarrow$ Summarization-2 as Evidence.

	Our Approach w/ various Deep Learning Classification methods								
	BERT$_{base}$			RoBERTa$_{base}$			XLNet$_{base}$		
	-	S1	S2	-	S1	S2	-	S1	S2
P	0.9612	0.9916	0.9917	0.9918	0.9929	0.9922	0.9920	0.9934	0.9947
R	0.9864	0.9888	0.9897	0.9897	0.9911	0.9916	0.9892	0.9911	0.9925
F	0.9858	0.9888	0.9893	0.9893	0.9908	**0.9908**	0.9892	0.9910	**0.9925**

Table 4: Results obtained after the fake news classification task where the results for each iteration of our approach with various deep learning classification methods are shown [P (Precision), R (Recall), and F (F-Score)]. (-) $\rightarrow$ No Evidence, S1 $\rightarrow$ Summarization-1 as Evidence, S2 $\rightarrow$ Summarization-2 as Evidence.

implementation with pre-trained GloVE[10] vectors, BERT$_{base}$, RoBERTa$_{base}$, and XLNET$_{base}$ -based classifiers. Our LSTM implementation uses *Adam* optimizer with a learning rate of 0.001, and 256 as the batch size. For classifiers based on BERT$_{base}$, RoBERTa$_{base}$, and XLNET$_{base}$, we use the HuggingFace implementations with a batch size of 32, L2 regularization and cross-entropy loss. The regularization parameter λ was set to 0.1. Each classification method is iterated **(1)** without evidence (-), **(2)** with augmented summarized evidences from S1, **(3)** and then with S2, thus giving us three sets of results for each method; as shown in Table 4.

As an input to the classifier, we use the claim as-is from the dataset as described above. We have a dataset $D = (x_n, y_n)_{n=1}^{N}$ comprising of N training samples. Here $x_n = (c_n, e_n)$, where c_n represents the claim, and e_n represents evidence gathered using our approach. $X \in \mathcal{X}$ is defined on input space, and $Y \in \mathcal{Y} = \{0, 1\}$ are the corresponding labels. Thus, given a claim c and evidence e, the aim of this task is to train a classifier such that the claim c is predicted as fake news or not, i.e $F_\theta : \mathcal{X} \rightarrow \mathcal{Y} \in \{0, 1\}$.

$$F(c, e; \theta) = \begin{cases} 1, & \text{if } c\, is\, the\, fake\, news \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where $F(c, e)$ is the function our model aims to learn over each iteration or epoch.

[10]https://nlp.stanford.edu/projects/glove/

6 Results and Discussion

The results for our classification task are shown in Table 4. Using our approach, we are able to marginally outperform (+0.24, F-Score) the previous state-of-the-art (SoTA) approaches for the task of fake news detection as shown in the last column (XLNet$_{base}$, S2). Even the RoBERT$_{base}$ model is able to outperform the SoTA approaches by a small margin. We present the values of our top two best models in boldface in Table 4. Although the improvement margin is small, we would like to note that the previous SoTA approaches are already performing at almost a 0.99 F-score. *We executed our model run multiple times to ensure that our improvement margin is indeed truly obtained.* We also observe that RoBERTA$_{base}$, and XLNet$_{base}$ outperform the SoTA approaches (Chen et. al. / Li et. al.) even with S1 summarization component. Classical machine learning-based approaches are also shown to perform very well for this task as the scores of 0.96 can be considered to be a good performance for any classification method.

However, this is not the only key takeaway from these results. *We observe that by using our novel approach, a consistent improvement is seen in the task results.* The efficacy of our approach can be seen from Table 4, as either S1 or S2 consistently outperforms all the base models (-) [no evidence] in the table. Moreover, using our approach, we are able to gather key evidence for such a dataset

Text	XLNet	LR	SVM
We always appreciate questions about the quality of our data. If you see a number that doesn't look right please file an issue at and we will investigate. [SEP] SOURCES: github.com	✗	✓	✓
The number of daily tests has been increasing in a steep climb. Average daily tests during the past three weeks also strongly depict the progress made in enhancement of #COVID19 tests across the country. [SEP] SOURCES: twitter.com/MoHFW_INDI	✗	✓	✓
Bill Gates said thousands of people will die with the COVID-19 vaccine [SEP] SOURCES:	✗	✗	✗

Table 5: Qualitative error analysis of some output cases both in terms of successes and failures of our approach.

where, to begin with, only claims were present with manually annotated labels. Table 5 illustrates the success and failure cases from XLNet, Logistic Regression and Support Vector Machines. We observe that first two cases were incorrectly predicted by the XLNet but were predicted correctly by LR and SVM. Last case was predicted incorrectly by all of the models. We believe that the absence of source in the text could be a potential reason for this failure. Our approach can gather the evidence using a fully automated method with summarization component(s) in the pipeline. The importance of this component can be gathered from manual observations of examples in the augmented dataset. We observe that summarized evidences shorten the length of the evidences, which helps the Transformer architecture-based classifiers like BERT$_{base}$, RoBERTa$_{base}$, XLNet$_{base}$ perform better. These pre-trained models have a token length limitation of 512 tokens which is easily able to capture our summarized evidence. We also manually observe that the summarization component helps reduce redundancy in the generated sentences and removes duplicates. Hence, improving the quality of evidence used as additional input helps reduce the training time. The performance of our models with the fine-tuned summarization component (S2) seems to perform better than S1, and the model without any evidence, as can be seen in Table 4.

We acknowledge that the CONSTRAINT dataset is saturated in terms of possible improvements. However, with this paper, *our aim is to show the efficacy of our summarization technique which can help the evidence detection for news*. We chose this dataset at an early stage of our work, and our experiments do show that improvements can, in fact, still be shown on this dataset. Our best-performing system surpasses the state-of-the-art by 0.23% points.

7 Conclusion and Future Work

In this paper, we present an automated method to collect evidence for the fake news detection task. We use our novel approach to augment the dataset, released in the CONSTRAINT-2021 Shared Task, with evidence sets collected from the web. Our method helps process these evidence sets, clean them and use them to generate summarized evidence based on two different methodologies. We use either of the summarized evidence as an additional input to the fake news classification task and perform an evaluation of our approach. We discuss the results of the classification task and conclude that our approach helps outperform the previous SoTA approaches by a small margin, however, helping generate evidence for a crucial dataset. We show that a summarization module can help collect evidence more effectively. We augment this dataset with the summarized evidence and release it along with the code and generated models for further research. We would also like to conclude that our method is generalizable; since it uses pre-trained metrics (BERTScore) and models (T5), it can be used to gather evidence for other datasets. The overall pipeline is also not very time-consuming (2 seconds per sample) once fine-tuned models are included in it. We hope our method and the resources are helpful to the NLP community.

In future, we would like to use our method to gather evidence for other fact detection/verification datasets as well. Our initial aim is to reproduce this study with other datasets and ensure that our method performs well in a real-world scenario. We would also like to apply this method and gather further evidence for existing fake news datasets, and perform our experiments to evaluate this approach over multiple exisiting datasets, including existing multilingual datasets.

References

Jennifer Allen, Baird Howland, Markus Mobius, David Rothschild, and Duncan J Watts. 2020. Evaluating the fake news problem at the scale of the information ecosystem. *Science Advances*, 6(14):eaay3539.

Sumitra Badrinathan. 2020. Educative interventions to combat misinformation: Evidence from a field experiment in india. *American Political Science Review*, pages 1–17.

Ben Chen, Bin Chen, Dehong Gao, Qijin Chen, Chengfu Huo, Xiaonan Meng, Weijun Ren, and Yang Zhou. 2021. Transformer-based language model fine-tuning methods for covid-19 fake news detection. In *Combating Online Hostile Posts in Regional Languages during Emergency Situation*, pages 83–92, Cham. Springer International Publishing.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada. 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.

Anna Glazkova, Maksim Glazkov, and Timofey Trifonov. 2020. g2tmn at constraint@aaai2021: Exploiting CT-BERT and ensembling learning for COVID-19 fake news detection. *CoRR*, abs/2012.11967.

Andreas Hanselowski, Hao Zhang, Zile Li, Daniil Sorokin, Benjamin Schiller, Claudia Schulz, and Iryna Gurevych. 2018. UKP-athene: Multi-sentence textual entailment for claim verification. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 103–108, Brussels, Belgium. Association for Computational Linguistics.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*, pages 1693–1701.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Debanjana Kar, Mohit Bhardwaj, Suranjana Samanta, and Amar Prakash Azad. 2020. No rumours please! A multi-indic-lingual approach for COVID fake-tweet detection. *CoRR*, abs/2010.06906.

Simon Kemp. 2015. Global digital & social media stats: 2015. *Social Media Today*.

Xiangyang Li, Yu Xia, Xiang Long, Zheng Li, and Sujian Li. 2021. Exploring text-transformers in aaai 2021 shared task: Covid-19 fake news detection in english. In *Combating Online Hostile Posts in Regional Languages during Emergency Situation*, pages 106–115, Cham. Springer International Publishing.

Christopher Malon. 2018. Team papelo: Transformer networks at FEVER. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 109–113, Brussels, Belgium. Association for Computational Linguistics.

Martin Müller, Marcel Salathé, and Per Egil Kummervold. 2020. Covid-twitter-bert: A natural language processing model to analyse COVID-19 content on twitter. *CoRR*, abs/2005.07503.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *CoRR*, abs/1606.01933.

Parth Patwa, Shivam Sharma, Srinivas PYKL, Vineeth Guptha, Gitanjali Kumari, Md Shad Akhtar, Asif Ekbal, Amitava Das, and Tanmoy Chakraborty. 2020. Fighting an infodemic: Covid-19 fake news dataset.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

Tathagata Raha, Vijayasaradhi Indurthi, Aayush Upadhyaya, Jeevesh Kataria, Pramud Bommakanti, Vikram Keswani, and Vasudeva Varma. 2021. Identifying COVID-19 fake news in social media. *CoRR*, abs/2101.11954.

Arjun Roy, Kingshuk Basak, Asif Ekbal, and Pushpak Bhattacharyya. 2018. A deep ensemble framework for fake news detection and classification. *CoRR*, abs/1811.04670.

Gautam Kishore Shahi and Durgesh Nandini. 2020. Fakecovid - A multilingual cross-domain fact check news dataset for COVID-19. *CoRR*, abs/2006.11343.

S. M. Sadiq-Ur-Rahman Shifath, Mohammad Faiyaz Khan, and Md. Saiful Islam. 2021. A transformer based approach for fighting COVID-19 fake news. *CoRR*, abs/2101.12027.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on socialwang media: A data mining perspective.

Elena Shushkevich and John Cardiff. 2021. TUDublin team at Constraint@AAAI2021 – COVID19 Fake News Detection. *arXiv e-prints*, page arXiv:2101.05701.

Sabrina Tavernise. 2016. As fake news spreads lies, more readers shrug at the truth. *The New York Times*, 6.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018a. 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, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. 2018b. The fact extraction and verification (fever) shared task.

James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. 2019. The FEVER2.0 shared task. In *Proceedings of the Second Workshop on Fact Extraction and VERification (FEVER)*, pages 1–6, Hong Kong, China. Association for Computational Linguistics.

Rutvik Vijjali, Prathyush Potluri, Siddharth Kumar, and Sundeep Teki. 2020. Two stage transformer model for COVID-19 fake news detection and fact checking. In *Proceedings of the 3rd NLP4IF Workshop on NLP for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 1–10, Barcelona, Spain (Online). International Committee on Computational Linguistics (ICCL).

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 18–22, Baltimore, MD, USA. 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 (Volume 2: Short Papers)*, pages 422–426, Vancouver, Canada. Association for Computational Linguistics.

Apurva Wani, Isha Joshi, Snehal Khandve, Vedangi Wagh, and Raviraj Joshi. 2021. Evaluating deep learning approaches for covid19 fake news detection. *CoRR*, abs/2101.04012.

Herman Wasserman and Dani Madrid-Morales. 2019. An exploratory study of "fake news" and media trust in kenya, nigeria and south africa. *African Journalism Studies*, 40(1):107–123.

Prediction of Video Game Development Problems Based on Postmortems using Different Word Embedding Techniques

Anirudh A[1], AMAN RAJ SINGH[2], Anjali Goyal[3], Lov Kumar[4], N L Bhanu Murthy[5]
BITS Pilani Hyderabad[1,2,4,5]
AMITY University[3]
(f20180936[1],f20191483[2],lovkumar[4],bhanu[5])@hyderabad.bits-pilani.ac.in
anjaligoyal19@yahoo.in[3]

Abstract

The interactive entertainment industry is being actively involved with the development, marketing and sale of video games in the past decade. The increasing interest in video games has led to an increase in video game development techniques and methods. It has emerged as an immensely large sector, and now it has grown to be larger than the movie and music industry combined. The postmortem of a game outlines and analyzes the game's history, team goals, what went right, and what went wrong with the game. Despite its significance, there is little understanding related to the challenges encountered by the programmers. Postmortems are not properly maintained and are informally written, leading to a lack of trustworthiness. In this study, we perform a systematic analysis on different problems faced in the video game development. The need for automation and ML techniques arises because it could help game developers easily identify the exact problem from the description, and hence be able to easily find a solution. This work could also help developers in identifying frequent mistakes that could be avoided, and will provide researchers a beginning point to further consider game development in context of software engineering.

1 Introduction

The video game industry is engaged in the process of development, promotion, and selling video games. It includes several occupation disciplines and employs a huge number of individuals across the globe. The business has developed from focused markets to the mainstream in recent years. Despite being an extremely competitive market where knowledge is the principle weapon, absence of information regarding processes and techniques used in game development makes it hard to understand the game development process. Due to this, developers often find it difficult to avoid commonly occurring issues and learn from past faults.

The motive behind this work is to classify the dataset based on the quote into types of problems so that future developers could easily recognize the type of problem they are facing and find solution accordingly. However, there are 3 main challenges in this process:

- ***Word Embedding:*** The post-mortems of game development are not well structured (Washburn Jr et al., 2016). This poses an intrinsic challenge. Since the input to any machine learning (ML) model is a feature vector, it is crucial to give a numerical representation of the textual data. This challenge can be reduced by using word embedding techniques. Word embedding techniques not only give a numerical representation of the textual data, but also combine the words with similar meaning and provide a reduced set of features (Li and Yang, 2018). In this work, 7 word embedding techniques - TFIDF, Skip gram, CBOW, Word2Vec, BERT, GloVe, and FastText are applied on the text and their predictive abilities are compared.

- ***Number of Features:*** The efficiency of any ML model relies up on its features. Research (Cai et al., 2018) suggests that models where the input consisted of redundant and irrelevant features performed less efficiently. Since the data set consists of a huge number of features, this poses an intrinsic challenge. To overcome this, 3 feature selection techniques have been used to select the relevant and crucial features: Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA) and Analysis of variance (ANOVA).

- ***Class Imbalance:*** A balanced dataset (Junsomboon and Phienthrakul, 2017) is one that contains an equal or almost equal number of samples from all the dependent variables. The last challenge in building the predictive model is that the data suffers from class imbalance problem. Hence, Synthetic Minority Oversampling Technique (SMOTE) has been used to balance the data.

To overcome the 3 challenges, a technical analysis and comparative study between the performance of 7 word embedding, 3 feature selection, and 5 classification techniques have been conducted. 5 classification techniques namely, K-Nearest Neighbours (KNN), Support Vector Classifier (SVC), Naive Bayes Classifier (NBC), Decision Tree (DT) and Random Forest (RF) have been used. Finally, the performance of the word embedding and classification techniques are compared. The results obtained from the original data is also compared to the results obtained from SMOTE data. To compare the results, accuracy, F-measure, and area under the curve (AUC) are used. Box plots are drawn based on AUC values since accuracy is not a good measure when the data suffers from class imbalance. Finally, rank sum test and friedman's test are used to test the hypotheses.

2 RELATED WORK

This section details about studies available in the broad domain of game development. (Politowski et al., 2020) prepared a grounded dataset obtained from post-mortems of video games which details about various software engineering problems during development. An iterative method has been used to create the dataset. 1035 problems were extracted from more than 200 post-mortems spanning over 20 years (1998-2018). The problems are divided into 3 problem groups (production, business, management) which are further divided into 20 different types. This work utilizes the above stated dataset to understand issues encountered by developers during the process of video-game development and further we foster a model for distinguishing the problem group based on description.

Game industry problems: An extensive analysis of the gray literature (Politowski et al., 2021) tries to analyse and develop a state of the problems of the gaming industry, their evolution and root causes which would help researchers and practitioners to work towards addressing and solving these problems. It was observed that the industry suffers from similar proportions of management and production related problems. Over the years as the industry became more mainstream, management related problems decreased only to give space to business related problems. Technical and design related problems have also decreased over the years. Team related problems increased over the last decade, and marketing problems had the biggest increase over the past 23 years. Finally, it was concluded that people (and not technology) were the root cause of most problems.

Callele et al. (Callele et al., 2005) analysed the various factors which led to the success or failure of a video game. Using the Game Developer Magazine, they analysed 50 post-mortems and investigated how requirements engineering could be applied to game development. "What went right" and "What went wrong" aspects were grouped into 5 categories: (1) pre-production issues (2) internal, and management related problems (3) external problems (4) technological problems (5) scheduler related problems. Finally, it was concluded that the transition from preproduction to production plays a crucial role in deciding the fate (success/failure) of the developed video game.

20 post-mortems were taken from the Gamasutra Website and were analysed by Petrillo et al. (Petrillo et al., 2009) The most common game development problems were identified and they were compared with traditional software-engineering problems. It was concluded that (1) management (and not technical) related issues contribute most to the video-game development problems (2) problems faced in video-game development and traditional software development are very similar and, (3) the most common problems are related to scope, feature creep, and cutting features.

Washburn et al. (Washburn Jr et al., 2016) analysed 155 game development post-mortems for what went right and wrong. Various characteristics of game development have been identified, linked with positive and negative experiences and a set of best practices, pitfalls for game development have been distilled. Design aspects cover all situations and decisions that were made that are external to the direct team and development process. Production issues relate to scheduling and work prioritizing issues. Other aspects include art, programming and testing issues.

3 Study Design

This section enlists the information about different design plans utilized in this work.

3.1 Experimental Dataset

This work uses Video Game Development problems data set which was collected from MSR 2020 conference datasets (Politowski et al., 2020). The dataset was created using iterative method, where more than 200 post-mortems for different video games were studied and around 1035 problems related to software engineering were extracted. The problems were divided into 3 groups – production, management, and business (Politowski et al., 2021). Production problems can be classified as problems based on documentation, prototyping, technical, testing, tools, bugs, and design. Business problems could be due to marketing or monetization, while management problems could be classified as problems based on communication, crunch time, delays, team, budget, planning, security, scope, cutting features, feature creep, and multiple projects.

3.2 Word Embedding

The representation of words for text analysis, in the form of a real-valued vector is called word embedding. The feature vectors encode meaning of the word such that words with similar meaning are closer in vector space. This reduces the overall feature space. Primarily, there are 2 word embedding methods: frequency-based, and neural network-based. In this work, 7 word embedding techniques have been applied to represent the words as a vector in n-dimensional vector space. The data has been cleaned by removing stop-words, bad symbols, spaces, etc. Further, predictive power of word embedding techniques have been compared.

3.3 SMOTE

The considered dataset suffers from the problem of class imbalance. Out of the four categories, the maximum class has around 430 data points while the minority class has less than 100 data points. Since ML algorithms increase accuracy by reducing the error, the class distribution is not considered. This problem is also prevalent in various domains such as fraud or anomaly detection, face identification, etc. Conventional ML algorithms such as logistic regression, DT, etc. possess bias towards majority class (Hoens and Chawla, 2013). Hence, dataset is balanced using SMOTE technique (Fernández et al., 2018)(Chawla, 2009). SMOTE balances the class distribution by replicating minority class instances.

3.4 Feature Selection

This work utilizes 3 feature selection techniques: ANOVA, PCA, and LDA for eliminating irrelevant features. The predictive power of the classifiers learnt using selected features is compared with the predictive power of the classifiers learnt using all features using AUC, F-measure, and accuracy. Further, rank sum test has been applied.

- ***ANOVA*** is a collection of statistical models for analysing differences among means (Sarstedt and Mooi, 2019)(St et al., 1989). The one-way classification follows completely random design (CRD), while two-way classification follows random block design (RBD). Overall, ANOVA possess no assumptions. However, CRD assumes independence, normality and homogeneity of variances of the residuals while RBD assumes homogeneity of variances of residuals. ANOVA partitions total sum of squares (SS) into components related to the effects used in model. For instance, model for a simplified ANOVA with one type of treatment at different levels would have $SS_{Total} = SS_{Error} + SS_{Treatments}$. For comparing factors of total deviation, below formula for F-test is used:

$$F = \frac{Variance\ between\ treatments}{Variance\ within\ treatments} \tag{1}$$

- ***PCA*** is an unsupervised dimensionality-reduction technique to transform large number of features into a smaller set which comprises similar information as contained by large number of features. Each instance is projected onto only principal components to reduce dimensionality while preserving data variation. If variance is high, it is easier to find patterns in the data set and so, we choose the ones with high variance as the important features.

- ***LDA*** is a supervised learning technique for dimensionality reduction where classes and their dependencies are also considered (Martinez and Kak, 2001)(Yu and Yang, 2001). Unlike PCA which considers maximum variance alone, LDA considers within class and between class variance also. The objective of

LDA is to extend components from higher dimensional space onto a lower dimensional space to keep away from the curse of dimensionality and furthermore decrease resources and dimensional expenses. Reducing the dimensions shrinks and concludes the dimensions which helps in better understanding of the data. LDA measures data from all features to make a new axis that limits variance and maximizes class distance.

3.5 Classification Techniques

The performance of various word-embedding, feature selection and SMOTE is evaluated with 5 ML classifiers: KNN, SVC, NBC, DT, & RF.

4 RESEARCH METHODOLOGY

The motive behind this work is to do a technical analysis and comparison between performance of 7 word embedding and 5 classification techniques on game development problems. The algorithm tries to classify the dataset based on the quote into types of problem so that future developers could easily recognize the type of problem they are facing and find solution accordingly. Firstly, 7 word embedding techniques are applied on text available in the dataset to obtain feature vectors. Next, data imbalance problem was dealt with using SMOTE technique which adds more minority test cases and balances data. Dimensionality reduction and feature selection was done using PCA, LDA and ANOVA. Finally, 5 classification techniques were used to predict class of the test data using embedded vectors. The performance of various word embedding and classification techniques are compared. The results obtained from original data are also compared to that obtained from SMOTE data using accuracy, F-measure and AUC. Box plots are drawn based on the AUC values since accuracy is not a good measure when the data suffers from class imbalance (Bekkar et al., 2013). Rank sum test and Friedman's test are used to test the hypotheses. Framework of the proposed work is depicted in Figure 1.

5 Empirical Results and Analysis

In this work, 7 word embedding, 1 sampling, 3 feature selection, and 5 ML classifiers were applied to develop models which predict the group of game development problem. Each word-embedding is applied on the chosen dataset and its effectiveness is evaluated using classifiers. The predicted values for each of the word embedding and classification techniques have been tabulated in Table 1. AUC, accuracy, and F-Measure values are calculated. To compare the results, however, only AUC values are used. This is because accuracy is not a good measure when there is class imbalance and AUC uses probability measures. Table 1 denotes AUC values corresponding to the original and SMOTE sampled data. Figure 2 shows the bar graph of AUC score for models trained on original data and balanced data for different sets of features. The models are validated using 5-fold cross validation (CV). AF denotes the AUC values corresponding to all features while ANOVA, PCA, LDA correspond to those got after feature selection. From Table 1 and Figure 2, we can infer following:

- High values of AUC confirm that developed models can predict different video game development problems based on the data.

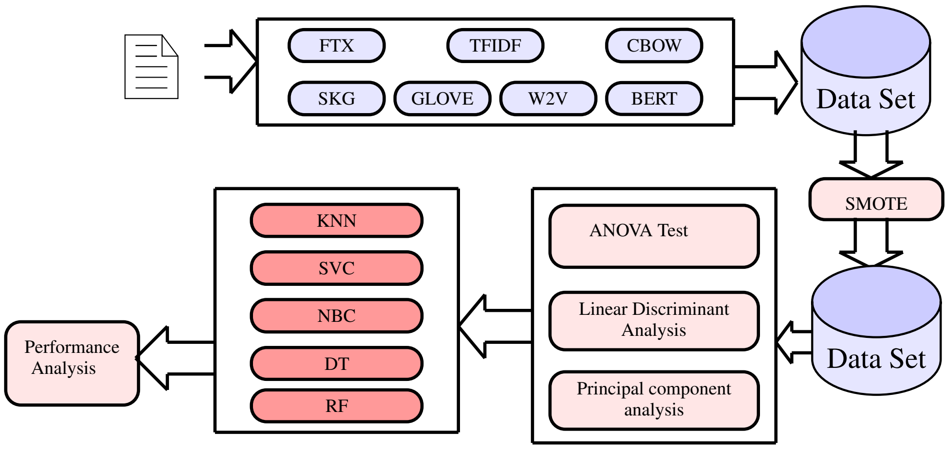
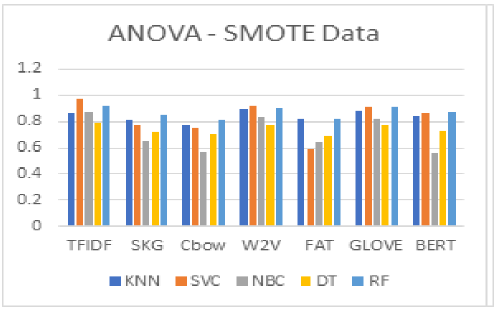


Figure 1: Research Framework

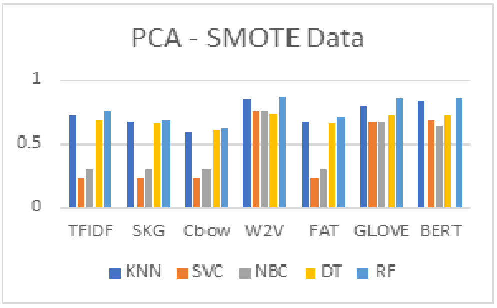
Table 1: AUC values

		Original Data					SMOTE Data				
		KNN	SVC	NBC	DT	RF	KNN	SVC	NBC	DT	RF
AF	TFIDF	0.54	0.82	0.53	0.62	0.70	0.83	0.99	0.96	0.80	0.92
	SKG	0.58	0.75	0.63	0.53	0.61	0.80	0.77	0.64	0.71	0.83
	Cbow	0.55	0.74	0.55	0.51	0.59	0.76	0.77	0.54	0.69	0.79
	W2V	0.79	0.82	0.80	0.59	0.70	0.89	0.93	0.83	0.74	0.90
	FAT	0.55	0.61	0.59	0.51	0.56	0.81	0.59	0.61	0.69	0.81
	GLOVE	0.78	0.82	0.80	0.59	0.74	0.89	0.91	0.82	0.76	0.91
	BERT	0.60	0.71	0.53	0.54	0.57	0.84	0.88	0.56	0.74	0.88
ANOVA	TFIDF	0.60	0.90	0.73	0.61	0.77	0.86	0.97	0.87	0.79	0.92
	SKG	0.57	0.75	0.63	0.54	0.63	0.81	0.77	0.65	0.72	0.85
	Cbow	0.54	0.74	0.55	0.53	0.59	0.77	0.75	0.57	0.70	0.81
	W2V	0.79	0.83	0.81	0.57	0.72	0.89	0.93	0.83	0.77	0.90
	FAT	0.55	0.60	0.62	0.51	0.56	0.83	0.59	0.64	0.69	0.83
	GLOVE	0.78	0.82	0.80	0.60	0.70	0.89	0.91	0.82	0.77	0.91
	BERT	0.59	0.71	0.54	0.53	0.57	0.84	0.86	0.56	0.73	0.87
PCA	TFIDF	0.59	0.44	0.50	0.52	0.57	0.73	0.22	0.30	0.69	0.76
	SKG	0.52	0.43	0.50	0.52	0.51	0.67	0.22	0.29	0.66	0.69
	Cbow	0.54	0.43	0.50	0.53	0.54	0.60	0.22	0.29	0.61	0.62
	W2V	0.68	0.77	0.77	0.60	0.70	0.85	0.76	0.76	0.74	0.87
	FAT	0.50	0.47	0.50	0.52	0.52	0.68	0.22	0.29	0.66	0.71
	GLOVE	0.63	0.73	0.72	0.57	0.66	0.80	0.68	0.67	0.73	0.85
	BERT	0.58	0.66	0.60	0.55	0.58	0.84	0.68	0.64	0.72	0.86
LDA	TFIDF	0.93	0.88	0.92	0.87	0.95	0.97	0.86	0.92	0.92	0.97
	SKG	0.94	0.97	0.97	0.87	0.95	0.97	0.97	0.97	0.91	0.98
	Cbow	0.94	0.96	0.97	0.84	0.94	0.96	0.97	0.97	0.89	0.97
	W2V	0.94	0.97	0.97	0.85	0.95	0.97	0.97	0.97	0.90	0.97
	FAT	0.78	0.84	0.84	0.66	0.78	0.87	0.84	0.84	0.78	0.89
	GLOVE	0.94	0.97	0.97	0.84	0.94	0.97	0.97	0.97	0.90	0.97
	BERT	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00	0.99	1.00

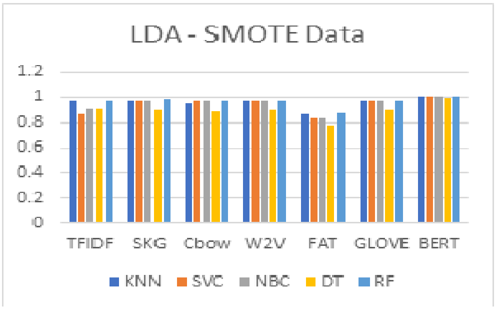
- Word2Vec embedding gives the best results.
- Results of FastText are poorer than others.
- Model trained using RF has better predictions.
- Model trained on SMOTE data has better AUC score than original data.



(2.1) AUC: ANOVA on SMOTE



(2.2) AUC: PCA on SMOTE



(2.3) AUC: LDA on SMOTE

Figure 2: AUC vlaue

6 Comparative Analysis

In this section, we compare the performance of models built using 7 word embedding, SMOTE, 3 feature selection, & 5 classification techniques. Descriptive statistics, box-plots, & significant tests have been used to compare developed models.

6.1 Word Embedding

In this work, 7 word embedding techniques (TFIDF, Skipgram, CBOW, Word2vec, FastText, GloVe and BERT) have been applied to represent words as vectors in n-dimensional vector space. The data has been cleaned before these techniques were applied, i.e., stop-words, bad symbols, spaces, etc. have been removed. These techniques not only give a numerical representation of textual data, but also encode their meaning such that words which are similar in meaning are closer in vector space. Moreover, as compared to a large set of vocabulary, a small number of features is obtained. The predictive ability of developed models using word embeddings are computed with the help of AUC score, F-Measure, and accuracy value. However, only AUC scores are considered for comparison since the data suffers from class imbalance. The AUC values are compared using descriptive statistics, box-plots, and significant tests.

6.1.1 Box-Plot: Word Embedding

Figure 3 provides descriptive statistics and performance values (measured using AUC) of 7 word embedding techniques in terms of a box-plot. From Figure 3, it is clear that models developed using Word2Vec, TFIDF, GloVe better predict group of game development problem. As compared to other techniques, models developed using CBOW, FastText, Skipgram have a low predictive ability. This is evident from the fact that mean AUC scores of CBOW, Skipgram & FastText are 0.66, 0.70 & 0.64 respectively while mean AUC scores of TFIDF,

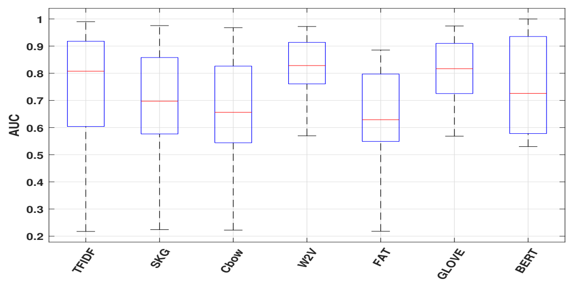


Figure 3: AUC Scores for Word Embedding Techniques

Word2vec & GloVe are 0.80, 0.83 & 0.82 respectively.

6.1.2 Significant Tests: Word Embedding

In this study, Friedman's test and rank-sum test are applied on the AUC scores to statistically compare the predictive ability and performance of the models built using 7 word embedding techniques. To check if these models have a significant improvement on the predictive ability or not, the following hypothesis has been formed:

- ***Null Hypothesis:*** The performance of the models do not depend on features extracted from word embedding techniques.
- ***Alternate Hypothesis:*** The performance of the models depend on features extracted from word embedding techniques.

To test the hypothesis, Friedman's test and rank sum test are used with a significance level of 0.05 i.e., null hypothesis is accepted if p≥0.05. For the purpose of simplicity a two-number representation of results has been used, i.e., 0 if null hypothesis is accepted (models are significantly same) and 1 if hypothesis is rejected (models are significantly different). From Table 2, it can be seen that Skipgram and CBOW give significantly different results as compared to Word2Vec, GloVe. Similarly, FastText gives significantly worse results as compared to TFIDF, GloVe, BERT.

Further, since models prove to give significantly different results, Friedman's mean rank test is also applied on AUC values to rank 7 word embedding models. A model with a lower mean rank value performs better than the one with a higher mean rank value. Hence, from Table 4 we can conclude that w2v gives the best results (followed by GloVe) and that its performance is significantly better than Skipgram, CBOW and FastText.

Table 2: Rank-Sum Test: Word Embedding

	Rank-Sum							**Friedman's**
	TFIDF	**SKG**	**CBOW**	**W2V**	**FAT**	**GLOVE**	**BERT**	**Mean-Rank**
TFIDF	0	0	0	0	1	0	0	3.32
SKG	0	0	0	1	0	1	0	4.30
CBOW	0	0	0	1	0	1	0	5.82
W2V	0	1	1	0	1	0	0	2.27
FAT	1	0	0	1	0	1	1	6.27
GLOVE	0	1	1	0	1	0	0	2.625
BERT	0	0	0	0	1	0	0	3.37

6.2 SMOTE

In this paper, it has been proposed to apply SMOTE in order to get balanced data and account for class imbalance. In this section, the predictive ability of prediction models using original data and SMOTE sampled data are compared using descriptive statistics, boxplot diagram, and significant tests.

6.2.1 Box Plots: Original and SMOTE data

The AUC values of the models developed using original data and SMOTE sampled data have been compared using box-plot diagrams as shown in Figure 4 and descriptive statistics. The information in Figure 4 shows that the models developed using SMOTE sampled data achieved 0.82 mean AUC score while that of original data achieved 0.64 mean AUC value. Hence, it can be concluded that SMOTE data sampling technique plays a crucial role in enhancing the model's ability to predict the group of game development problem.

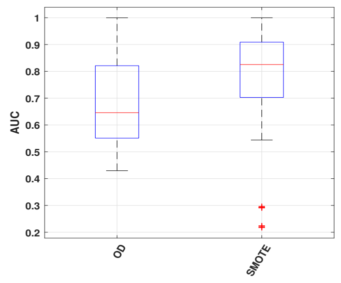


Figure 4: AUC Scores for Original data and SMOTE data

6.2.2 Significant Tests: Original and SMOTE data

In this study, Friedman's test and rank sum test are applied on the AUC scores to statistically compare the predictive ability of the models built using the original data and SMOTE sampled data. The objective is to check if the models developed using SMOTE has a significant improvement on the predictive ability or not, and for this, the following hypothesis has been formed:

Table 3: Significant Tests: Original and SMOTE data

Rank-Sum			Friedman's
	OD	SMOTE	Mean-Rank
OD	0	1	1.871
SMOTE	1	0	1.128

- ***Null Hypothesis (H_0)***: The models developed using SMOTE sampled data do not have a significant impact on the predictive ability of classification model.
- ***Alternate Hypothesis (H_a)***:The models developed using SMOTE sampled data have a significant impact on the predictive ability of classification model.

To test above hypothesis, p-value is used. Considering a significance level of 0.05 level (95% confidence interval), null hypothesis is accepted if the p-value≥0.05. From Table 3, it can be seen that the models built from the original data and SMOTE sampled data are significantly different. From Table 3, we can also conclude that the model developed using SMOTE sampled data performs significantly better and that accounting for class imbalance is crucial.

6.3 Feature Selection

In this work, 3 feature selection techniques were used to remove unnecessary features and for dimensionality reduction. The 3 feature selection techniques – ANOVA (Figure 22.1), PCA (Figure 22.2), and LDA (Figure 22.3) have been applied to find best combination of relevent features. The predictive ability of the developed models using the 3 feature selection techniques are evaluated using AUC scores. They are compared using descriptive statistics, boxplot diagram, and significant tests.

6.3.1 Box Plots: Different sets of Features

Figure 5 provides the descriptive statistics and performance values (measured using AUC) of the 3 feature selection techniques in terms of a box-plot. From Figure 5, it is evident that the models developed using LDA best predict the group of game development problem. The models developed using ANOVA, PCA have a relatively low predictive ability as compared to LDA. This is evident from the fact that the mean AUC scores ANOVA, PCA, LDA are 0.75, 0.6, 0.97 respectively.

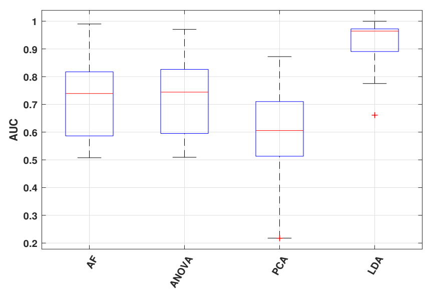


Figure 5: Feature Selection

Table 4: Significant Tests: Different sets of Features

Rank-Sum					Friedman's
	AF	ANOVA	PCA	LDA	Mean-Rank
AF	0	0	1	1	2.757
ANOVA	0	0	1	1	2.428
PCA	1	1	0	1	3.757
LDA	1	1	1	0	1.057

6.3.2 Significant Tests: Different sets of Features

In this study, Friedman's test and rank sum test are applied on the AUC scores to statistically compare the predictive ability of the models developed from 3 feature selection techniques. To test the hypothesis, p-value is used at significance level of 0.05 level (95% confidence interval), null hypothesis is accepted if the p-value > 0.05. For the purpose of simplicity, a two-number representation for the results has been used, i.e., 0 if the null hypothesis is accepted (models are significantly same) and 1 if the hypothesis is rejected (models are significantly different). From Table 4, it can be seen that the models developed using PCA, LDA are significantly different from the model developed using all features. Similarly, models developed using LDA is significantly different from all other models.

Further since models prove to give significantly different results, Friedman's mean rank test is performed on AUC values to rank the models developed using feature selection techniques. A model with a lower mean rank value performs better than one with a higher mean rank value. Hence, from Table 4 we can conclude that model developed using LDA gives best results while model developed using PCA gives worst prediction results. Also, it can be seen that models built using ANOVA, all features are not significantly different from one another. These results also verify the conclusion made from the box plot in Figure 5.

6.4 Classification Techniques

In this study, KNN, SVC, NBC, DT, RF have been used to classify the game development problem into 3 groups (production, management, and business-related problems). 5-fold CV has been used to train the prediction models. In this section, the predictive ability of the models developed using 5 classifiers are computed using AUC scores. The AUC values are compared using descriptive statistics, box-plots, & significant tests.

6.4.1 Box Plots: Classification Techniques

Figure 6 provides descriptive statistics and performance values (measured using AUC) of 5 classification techniques in terms of a box-plot. From Figure 6, it is clear that models developed using KNN, SVC, RF better predict group of game development problem. Compared to models built using KNN, SVC, and RF, models developed using NBC, DT have a low predictive ability. This is evident from the fact that mean AUC scores of KNN, SVC & RF are 0.80, 0.80 & 0.81 while mean AUC scores of NBC & DT are 0.70 & 0.69 respectively.

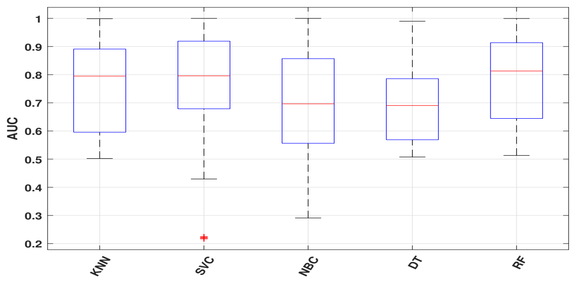


Figure 6: Classification Techniques

6.4.2 Significant Tests: Classification Techniques

In this study, Friedman's test and rank sum test are applied on AUC scores to statistically compare the predictive ability of models developed using 5 classifiers. Table 5 show results of Friedman's test and rank sum test for different techniques. For the purpose of simplicity, two-number representation for the results is used, i.e., 0 if null hypothesis is accepted (models are significantly same) and 1 if hypothesis is rejected (models are significantly different). From Table 5, it can be seen that model developed using DT is significantly different from models developed using KNN, SVC, RF while it does not differ significantly from NBC in terms of predictive ability. Further since the models prove to give significantly different results, Friedman's mean rank test is also applied on AUC values to rank 5 models. A lower value of mean rank indicates a better performance of the model. Hence, from Table 5, we can conclude that RF classifier gives the best results (followed by SVC, KNN) and its performance is significantly better than model built using DT.

Table 5: Significant Tests: Classification Techniques

	Rank-Sum					**Friedman's**
	KNN	**SVC**	**NBC**	**DT**	**RF**	**Mean-Rank**
KNN	0	0	0	1	0	3.000
SVC	0	0	0	1	0	2.375
NBC	0	0	0	0	0	3.107
DT	1	1	0	0	1	4.303
RF	0	0	0	1	0	2.214

7 Conclusion

A postmortem is a summarization procedure used to analyse the various positive and negative aspects of the game development project. It aids developers in drawing meaningful conclusions and helps them learn from past successes and failures. However, given the various responsibilities of a video game developer, it is not surprising that they hardly take time to conduct and prepare project postmortems. Moreover, the lack of formal structure leads to a lack of trust worthiness. In this work, a data set of the different problems in game development has been taken and studied. 7 word embedding techniques have been applied on the data set and it can be observed that Word2Vec gives the best results and these results are significantly different from Skipgram, CBOW and FastText. As far as the classification techniques are concerned, KNN, SVC, and RF produce significantly better results. RF gives the best value in Friedman's rank sum test. However, KNN may also be considered since it takes the lowest computation time amongst KNN, SVC, RF whilst still producing similar results. LDA is the best suitable feature selection technique here, followed by ANOVA. Finally, it is important to note that the data obtained after SMOTE gives significantly better yield than the original data, and hence, accounting for class imbalance is crucial. This work, thus provides a way to classify the dataset based on the quote into types of problems. This could help future developers easily recognize the type of problem they are facing and find suitable solutions.

References

Mohamed Bekkar, Hassiba Kheliouane Djemaa, and Taklit Akrouf Alitouche. 2013. Evaluation measures for models assessment over imbalanced data sets. *J Inf Eng Appl*, 3(10).

Jie Cai, Jiawei Luo, Shulin Wang, and Sheng Yang. 2018. Feature selection in machine learning: A new perspective. *Neurocomputing*, 300:70–79.

David Callele, Eric Neufeld, and Kevin Schneider. 2005. Requirements engineering and the creative process in the video game industry. In *13th IEEE International Conference on Requirements Engineering (RE'05)*, pages 240–250. IEEE.

Nitesh V Chawla. 2009. Data mining for imbalanced datasets: An overview. *Data mining and knowledge discovery handbook*, pages 875–886.

Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C Prati, Bartosz Krawczyk, and Francisco Herrera. 2018. *Learning from imbalanced data sets*, volume 10. Springer.

T Ryan Hoens and Nitesh V Chawla. 2013. Imbalanced datasets: from sampling to classifiers. *Imbalanced learning: Foundations, algorithms, and applications*, pages 43–59.

Nutthaporn Junsomboon and Tanasanee Phienthrakul. 2017. Combining over-sampling and under-sampling techniques for imbalance dataset. In *Proceedings of the 9th International Conference on Machine Learning and Computing*, pages 243–247.

Yang Li and Tao Yang. 2018. Word embedding for understanding natural language: a survey. In *Guide to big data applications*, pages 83–104. Springer.

Aleix M Martinez and Avinash C Kak. 2001. Pca versus lda. *IEEE transactions on pattern analysis and machine intelligence*, 23(2):228–233.

Fábio Petrillo, Marcelo Pimenta, Francisco Trindade, and Carlos Dietrich. 2009. What went wrong? a survey of problems in game development. *Computers in Entertainment (CIE)*, 7(1):1–22.

Cristiano Politowski, Fabio Petrillo, Gabriel C Ullmann, and Yann-Gaël Guéhéneuc. 2021. Game industry problems: An extensive analysis of the gray literature. *Information and Software Technology*, 134:106538.

Cristiano Politowski, Fabio Petrillo, Gabriel Cavalheiro Ullmann, Josias de Andrade Werly, and Yann-Gaël Guéhéneuc. 2020. Dataset of video game development problems. In *Proceedings of the 17th International Conference on Mining Software Repositories*, pages 553–557.

Marko Sarstedt and Erik Mooi. 2019. Hypothesis testing and anova. In *A Concise Guide to Market Research*, pages 151–208. Springer.

Lars St, Svante Wold, et al. 1989. Analysis of variance (anova). *Chemometrics and intelligent laboratory systems*, 6(4):259–272.

Michael Washburn Jr, Pavithra Sathiyanarayanan, Meiyappan Nagappan, Thomas Zimmermann, and Christian Bird. 2016. What went right and what went wrong: an analysis of 155 postmortems from game development. In *Proceedings of the 38th International Conference on Software Engineering Companion*, pages 280–289.

Hua Yu and Jie Yang. 2001. A direct lda algorithm for high-dimensional data—with application to face recognition. *Pattern recognition*, 34(10):2067–2070.

Multi-task pre-finetuning for zero-shot cross lingual transfer

Moukthika Yerramilli[*]
Amazon / Bangalore
mky@amazon.com

Pritam Varma[*]
Amazon / Bangalore
spv@amazon.com

Anurag Dwarakanath[*]
Amazon / Bangalore
adwaraka@amazon.com

Abstract

Building machine learning models for low resource languages is extremely challenging due to the lack of available training data (either un-annotated or annotated). To support such scenarios, zero-shot cross lingual transfer is used where the machine learning model is trained on a resource rich language and is directly tested on the resource poor language. In this paper, we present a technique which improves the performance of zero-shot cross lingual transfer. Our method performs multi-task pre-finetuning on a resource rich language using a multilingual pre-trained model. The pre-finetuned model is then tested in a zero-shot manner on the resource poor languages. We test the performance of our method on 8 languages and for two tasks, namely, Intent Classification (IC) & Named Entity Recognition (NER) using the MultiAtis++ dataset. The results show that our method improves IC performance in 7 out of 8 languages and NER performance in 4 languages. Our method also leads to faster convergence during finetuning. The usage of pre-finetuning demonstrates a data efficient way for supporting new languages and geographies across the world.

1 Introduction

Recent advances in Natural Language Processing include the development of language models trained in an unsupervised fashion on large amounts of data (Devlin et al., 2019), (Radford and Sutskever, 2018). These models were extended to a multilingual context by training on data from a number of languages (Devlin et al., 2019), (Conneau et al., 2019). These multilingual models were found to perform well in a zero-shot cross lingual transfer tasks - i.e. the multilingual model is fine-tuned for a task in a resource rich language (such as English) and is directly tested on a resource poor language (such as Swahili) (Schlinger, 2019a)

In this paper, we investigate the usage of multi-task pre-finetuning in zero-shot cross lingual tasks. Pre-finetuning is a step between pre-training and fine-tuning, where a pre-trained model is trained on additional supervised learning tasks with the aim to improve pre-trained representations (Aghajanyan et al., 2021)(Liu et al., 2019). Typically, these additional tasks are unrelated to the tasks that the model will finally be fine-tuned on. Past work on multi-task learning showed its usefulness in a monolingual setting. In our work, we extend the pre-finetuning concept to a multilingual setting with zero-shot cross lingual transfer.

Our method is demonstrated on the XLM-Roberta (Conneau et al., 2019) pre-trained model and uses 8 additional auxiliary tasks from the GLUE benchmark (Wang et al., 2018) for the multi-task pre-finetuning on English. The resulting model is then applied for the joint Intent-Classification & Named Entity Recognition tasks (IC-NER). We show cross lingual transfer by fine-tuning for IC-NER on English and directly test on 8 different languages in a zero-shot manner. We use the MultiAtis++ dataset (Xu et al., 2020) for the IC-NER data in English and 8 other languages for fine-tuning and testing. Our results bring out multiple insights. We find that the multi-task pre-finetuned model is better by 5.12% relative (or 391 absolute basis points (bps)) on average across all 8 languages for the IC task at early stages of fine-tuning. This shows the ability of multitask pre-finetuning to improve the learnt representations of the pre-trained model. The results however indicate that such out of the box improvement in pre-trained models is not seen in the NER task, where the performance at early stages of fine-tuning degrades by 10% relative (or 525 bps absolute). We also find that pre-finetuning has improved the performance

*Equal contribution

in monolingual setting where the results for both IC and NER for English have improved by 6.53% relative (or 549 bps absolute) and 11.86% relative (or 700 bps absolute) respectively.

Improving the ability of cross lingual transfer in a zero shot setting has large implications. Through our method, products using language models can improve machine learning support for new languages where sufficient training data is not available. For example, the Indian sub-continent has 179 languages and 544 dialects (Wikipedia contributors, 2021b) and building machine learning services to cater to all multilingual users is a daunting task since enough data, both in the form of annotated and un-annotated, is not available.

This paper is structured as follows. In Section 2, we present the related work in zero-shot cross lingual transfer and progress in multi-task pre-finetuning. Section 3 presents our method of improving zero-shot cross lingual transfer through the use of multi-task pre-finetuning. We present the results in Section 4 and conclude with directions for future work in Section 5.

2 Related work

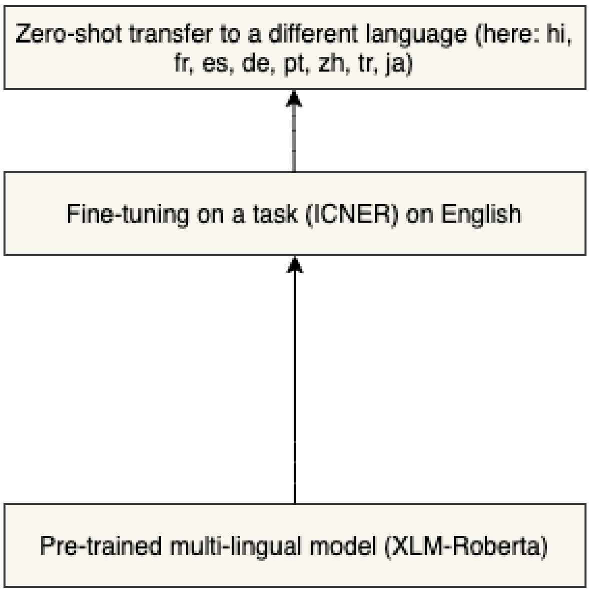
Multilingual language models such as mBERT (Devlin et al., 2019)(Devlin et al., 2021), XLM (Conneau et al., 2019) and MuRIL (Khanuja et al., 2021) have advanced the state-of-the-art on cross-lingual natural language understanding tasks by training large Transformer models (Vaswani et al., 2017) on data from many languages.

These multilingual pre-trained models can be used as generic task agnostic neural network architectures and can be applied in different natural language processing tasks by attaching a task specific decoder (such as a linear classifier) and fine-tuning on the task specific training data. The usage of multilingual pre-trained models also enabled zero-shot cross lingual transfer. In zero-shot transfer, the pre-trained model is fine tuned on annotated data from a resource rich language (such as English) and directly tested in a resource poor language (such as Swahili). Studies (Schlinger, 2019b) (Libovický et al., 2020a) (Wu and Dredze, 2019a) have shown such multilingual pre-trained models have the ability to learn common representations across languages even though the training methodology was not explicitly designed to build common representations.

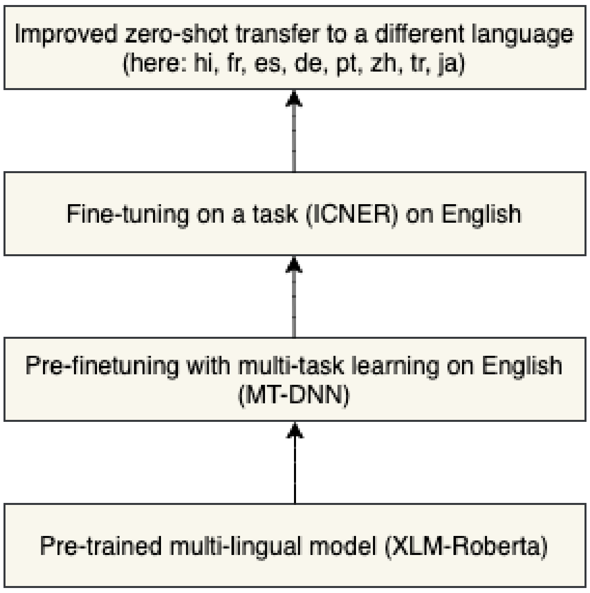
Independently, approaches have been developed to improve the performance of pre-trained models in the monolingual setting. Prominent work includes the usage of multi-task pre-finetuning (Liu et al., 2020) (Aghajanyan et al., 2021). In pre-finetuning, the pre-trained model such as BERT is trained in a supervised learning setting on auxiliary tasks. The work in (Liu et al., 2020) uses 8 different tasks from the GLUE dataset (Wang et al., 2018) and trains for different tasks in random batches. The work in (Aghajanyan et al., 2021) scales the pre-finetuning concept by training on 50 different tasks. Their results show that multi-task pre-finetuning can significantly improve the performance of the pre-trained models. Both these works test the improvement in a monolingual setting. In contrast, our work in this paper explores the applicability of pre-finetuning in a multilingual setting.

3 Multi-task pre-finetuning for zero shot cross lingual transfer

In this section, we present our method to improve the performance of zero-shot transfer of knowledge across languages. Current state-of-the-art methods have demonstrated the great ability of a neural network to transfer knowledge of a given task across languages using a pre-trained model that is trained generically (Conneau et al., 2018), (Schlinger, 2019a), (Wu and Dredze, 2019b). In our work, we demonstrate a further intriguing property of deep neural networks. We take a multi-lingual pre-trained neural network and train it over multiple tasks in English using task specific supervised data (we choose English since there is significant amount of supervised training data available for the language). We then fine tune the network for the specific down-stream task in English. We denote the resulting neural network model through our method as **MT-DNN-MultiLingual-Finetuned**. Now, when the **MT-DNN-MultiLingual-Finetuned** model is tested directly (i.e. zero-shot) on the downstream task in a different language, we observe better performance. This demonstrates the ability of the neural network to not only transfer knowledge of the same task across languages (as demonstrated by existing literature), but also shows its ability to transfer cumulative knowledge of multiple different tasks across languages. Figure 1 depicts the novelty of our work in comparison with extant literature.


Zero-shot transfer to a different language (here: hi, fr, es, de, pt, zh, tr, ja)
Fine-tuning on a task (ICNER) on English
Pre-trained multi-lingual model (XLM-Roberta)

(a) Baseline model: XLM-R with fine-tuning on English


Improved zero-shot transfer to a different language (here: hi, fr, es, de, pt, zh, tr, ja)
Fine-tuning on a task (ICNER) on English
Pre-finetuning with multi-task learning on English (MT-DNN)
Pre-trained multi-lingual model (XLM-Roberta)

(b) Experiment model: XLM-R with multi-task pre-finetuning and fine-tuning – both done on Enlgish

Figure 1: Baseline vs Experimental Model with multi-task pre-finetuning setup for zero-shot transfer.

3.1 Multi-Lingual Pre-trained Model

We base our method on the XLM-RoBERTa base (XLM-R) (Conneau et al., 2019) as the pre-trained model for all the experiments in this paper. XLM-R is trained on 100 languages (including the 8 languages we use for our tests) using the CommonCrawl Corpus. The XLM-R was trained using the Masked Language Model training objective and does not use any supervised data (or parallel corpus) for its training. Recent work (Schlinger, 2019a) (Libovický et al., 2020b) has shown that the XLM-R model is able to build similar representations for semantically similar sentences across languages.

3.2 The Multi-Task Pre-finetuning step

In existing work, the XLM-R model would be fine-tuned on a downstream task. In contrast, our method introduces a pre-finetuning step before fine-tuning is done. We chose the GLUE benchmark dataset (Wang et al., 2018) which contains supervised training data in English for 8 tasks spanning - 1) Single-Sentence Classification; 2) Pairwise Text Similarity; 3) Pairwise Text Classification; and 4) Pairwise Ranking. For each task, a task specific decoder was attached to the XLM-R pre-trained model and trained for a fixed number of epochs. We made use of the publicly available framework (Liu et al., 2020) for pre-finetuning on the GLUE benchmark. No layer freezing was done for the pre-finetuning steps. The different tasks and the size of the training data is shown in Table 1.

3.3 The finetuning step

Post the pre-finetuning on English using the GLUE benchmark dataset, we attach a decoder for the joint Intent Classification & Named Entity Recognition (IC-NER) task. The formulation of the IC-NER task is shown below.

$$y^i = softmax(W^i h_1 + b) \qquad (1)$$

where y^i represents the IC hypothesis and h_1 represents the hidden state of the classification head (also denoted using the special token [CLS]).

For NER, we feed the final hidden states of other tokens $h_2, ..., h_n$ into a softmax layer to classify over the slot filling labels. To make this procedure compatible with WordPiece or SentencePiece tokenization, we feed each tokenized input word into a tokenizer and use the hidden state corresponding to the first sub-token as input to the softmax classifier.

$$y^s_n = softmax(W^s h_n + b), n \in 1...N \qquad (2)$$

where h_n is the hidden state corresponding to the first sub-token of word x_n and s is the slot label. To jointly model intent classification and slot filling, the objective is formulated as:

$$p(y^i, y^s|x) = p(y^i|x) \prod_{n=1}^{N} p(y^s_n|x), \qquad (3)$$

Table 1: GLUE dev set results. ST-DNN has the same architecture as MT-DNN-Original but without the Multi-Task pre-finetuning. MT-DNN-MultiLingual is the Multi-Task pre-finetuned model with XLM-R pre-trained model as the base. The results in columns 3, 4 and 5 (BERT-Large, ST-DNN and MT-DNN-Original) are obtained from (Liu et al., 2019) The results in bold(not indicative of best performance) represent the performance of MT-DNN-MultiLingual, which is the pre-finetuned model used in experiments later.

Dataset	Train Data Size	BERT-Large	ST-DNN	MT-DNN-Original	MT-DNN-MultiLingual
QQP(F1score/Acc)	364k	86.3/86.2	91.3/88.4	91.9/89.2	**88.532/91.373**
MNLI-m/mm(Acc)	393k	86.3/86.2	86.6/86.3	87.1/86.7	**84.035/84.123**
RTE(Acc)	2.5k	71.1	72	83.4	**77.978**
QNLI(Acc)	108k	92.4	-	92.9	**90.427**
MRPC(F1/Acc)	3.7k	89.5/85.8	89.7/86.4	91.0/87.5	**92.199/89.216**
SST-2(Acc)	67k	93.5	-	94.3	**92.775**
CoLA(Mcc)	8.5k	61.8	-	63.5	**49.217**
STS-B(Pc/Sc)	7k	89.6/89.3	-	90.7/90.6	**89.086/88.868**

.

The learning objective is to maximize the conditional probability $p(y^i, y^s|x)$. The model is finetuned end-to-end via minimizing the cross-entropy loss.

3.4 Zero-shot transfer evaluation task

We evaluate the performance of pre-finetuned and finetuned XLM-R model (on English) directly on different languages for the joint IC-NER task. We use 8 languages in the MultiATIS++ corpus - Hindi (hi), French (fr), Spanish (es), German (de), Portuguese (pt), Chinese (zh), Japanese (ja) and Turkish (tr). These languages also belong to a diverse set of language families - Indo-European, Sino-Tibetan, Japonic and Altaic.

4 Experiments and Results

4.1 Pre-Finetuning

Using the XLM-R pre-trained model, pre-finetuning is conducted over eight English language GLUE datasets: CoLA(Single-Sentence Classification), SST-2(Single-Sentence Classification), STS-B (Pairwise Text Similarity), RTE(Pairwise Text Classification), MNLI(Pairwise Text Classification), QQP(Pairwise Text Classification), MRPC(Pairwise Text Classification), QNLI(Pairwise Ranking). Pre-finetuning is carried out using MT-DNN (Multi-task DNN) setup where the training is done on only English. The training is conducted for four epochs with standard (as per the original MT-DNN implementation(Liu et al., 2020)) hyper-parameter values such as learning rate of $5 \times e^{-5}$, batch-size of 32 and with adamax optimizer.

Table 1 shows the GLUE dev set results, where MT-DNN-MultiLingual is the proposed model with pre-finetuning over XLM-R. The MT-DNN-Original is the original MT-DNN model that is pre-finetuned on BERT-Large model. The results indicate that MT-DNN-MultiLingual, which uses a multi-lingual pre-trained model as its base, is able to beat the mono-lingual non-prefinetuned models such as ST-DNN (stands for single task DNN which implements task-wise finetuning) and BERT-Large in most validation datasets (except for CoLA, MNLI-m/mm). However, its unable to beat MT-DNN-Original which is pre-trained and pre-finetuned exclusively on English. These results re-emphasize the effectiveness of multilingual models even for mono-lingual tasks.

4.2 IC-NER fine-tuning and Zero-Shot Transfer

The baseline **XLM-R-Finetuned** model for zero-shot transfer experiments consists of XLM-R model that is fine-tuned on English and evaluated on other languages in a zero-shot manner. These experiments are conducted on IC-NER Joint Task (Chen et al., 2019) over MutiATIS++ dataset. Our method **MT-DNN-MultiLingual-Finetuned** consists of MT-DNN-MultiLingual that is fine-tuned on English. Tables 2, 3 and 4 show the zero-shot performance of baseline model vs **MT-DNN-MultiLingual-Finetuned** across all the available languages in MultiAtis++ dataset. The results are averaged across three different fine-tuning runs and hyper-parameters such as batch-size (256), learning rate ($5 \times e^{-6}$) remain the same across both baseline and experimental model.

As seen in the results, **MT-DNN-MultiLingual-Finetuned** beats the baseline on 7 out of 8 languages at the 10th epoch for the IC task. The average improvement in accuracy is 5.12% con-

Table 2: XLM-R-finetuned vs MT-DNN-MultiLingual-finetuned for English, Hindi and French

Language		en		hi		fr	
Task		IC (Accuracy)	NER (F1 score)	IC (Accuracy)	NER (F1 score)	IC (Accuracy)	NER (F1 score)
XLM-R-finetuned	10th Epoch	78.52	59	72.57	55	72.14	46
	20th Epoch	86.52	81	78.4	58	80.82	55
	30th Epoch	90.06	83	80.68	54	82.3	59
	40th Epoch	90.52	83	81.37	55	84.58	59
MT-DNN-MultiLingual-finetuned	10th Epoch	84.01	66	77.71	53	80.36	49
	20th Epoch	90.41	82	83.77	55	86.64	57
	30th Epoch	90.75	83	84.68	54	86.64	58
	40th Epoch	90.75	84	84.57	53	86.3	58

Table 3: XLM-R-finetuned vs MT-DNN-MultiLingual-finetuned for Spanish, German and Portuguese

Language		es		de		pt	
Task		IC (Accuracy)	NER (F1 score)	IC (Accuracy)	NER (F1 score)	IC (Accuracy)	NER (F1 score)
XLM-R-finetuned	10th Epoch	72.34	47	72.34	47	73.6	54
	20th Epoch	81.05	69	80.22	53	80.57	62
	30th Epoch	84.7	70	84.68	57	86.17	63
	40th Epoch	84.81	70	88.68	58	87.88	64
MT-DNN-MultiLingual-finetuned	10th Epoch	75.68	51	83.08	48	74.62	54
	20th Epoch	87.1	69	88.57	58	82.62	62
	30th Epoch	87.67	72	88.91	59	83.42	63
	40th Epoch	87.44	72	89.48	59	83.2	63

Table 4: XLM-R-finetuned vs MT-DNN-MultiLingual-finetuned for Chinese, Japanese and Turkish

Language		zh		ja		tr	
Task		IC (Accuracy)	NER (F1 score)	IC (Accuracy)	NER (F1 score)	IC (Accuracy)	NER (F1 score)
XLM-R-finetuned	10th Epoch	72.14	61	72.03	52	73.6	54
	20th Epoch	75.57	65	73.07	56	68.42	51
	30th Epoch	79.68	67	77.33	55	72.68	52
	40th Epoch	79.79	67	78.82	55	74.82	51
MT-DNN-MultiLingual-finetuned	10th Epoch	75.57	64	73.53	33	71.55	22
	20th Epoch	85.5	66	84.11	43	71.4	27
	30th Epoch	85.38	67	85.5	45	68.7	27
	40th Epoch	85.38	0.67	85.5	45	66.7	27

sidering all languages. The results indicate Turkish (tr) is showing regressions for both IC and NER and appears to be an outlier. Discounting the Turkish language, we see an average improvement of 6.11% for IC. A similar improvement of 6.54% in IC is seen on English. These results, at the 10^{th} epoch, indicate that multi-task pre-finetuning has improved the performance of the pre-trained model for the IC task in a zero-shot setting. At the 40^{th} epoch, we see that the performance improvement tapers and achieves an average improvement of 1.17% across all languages. This reduction in gain is expected as the continual training on English starts to improve baseline performance but reduces some of the gains across other languages - i.e. the usage of pre-finetuning allows for early convergence (convergence in terms of performance on non-English languages). Discounting the results from Turkish, we see an average improvement of 2.64% across 7 languages for the IC task at the 40^{th} epoch.

The early convergence of our method can be seen in figure 2. We observe that MT-DNN-MultiLingual-Finetuned converges faster (at epoch 20) than the baseline method. We also see that in Chinese, the baseline model appears to have a constant test accuracy value till 15 epochs and the accuracy starts to increase post that. In contrast, MT-DNN-MultiLingual-Finetuned accuracy starts to improve after 8 epochs. Since MT-DNN-MultiLingual-Finetuned gains such early momentum on IC task, its able to progressively beat the baseline performance. Similar early gains in accuracy are observed for Hindi and Japanese as well. This can be attributed to the improved generalisation via the pre-finetuning step.

We see a degradation of 5.32% and 12.17% for pt (Portuguese) and tr (Turkish) respectively on IC task (at 40th epoch). For Portuguese, we see that MT-DNN-MultiLingual-Finetuned beats the baseline until 25 epochs. However, fine-tuning for further epochs shows better gains in baseline model compared to MT-DNN-MultiLingual-Finetuned. Although the MT-DNN-MultiLingual-Finetuned beats the baseline for zero-shot performance in Turkish at 20th Epoch, the performance does not

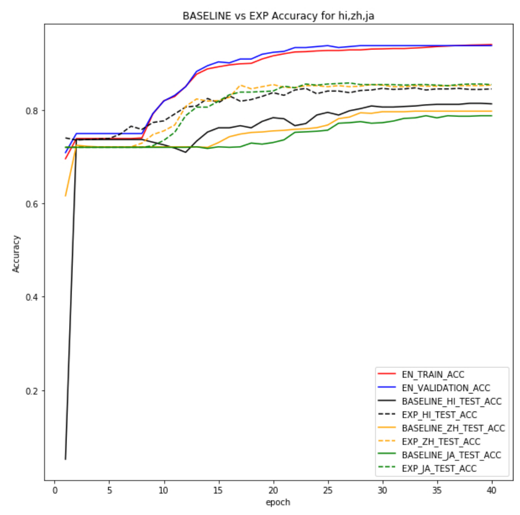


Figure 2: Learning profiles of Train (en), Validation (en) and Test (hi,zh,ja) datasets for Baseline and Experimental Models

improve with further finetuning as compared to baseline.

The results from NER indicate that pre-finetuning hasn't improved the performance. On average across all 8 languages, NER performance at the 10^{th} epoch has degraded by 10.09% relative to baseline. For NER, we see significant degradation of Turkish (- 47%) (- indicates degradation). Discounting Turkish, we see NER performance averaging across 7 languages to be -2.72% relative to baseline.

The reasons for degradation in Turkish (for both IC & NER) and for the lack of improvement in NER is not exactly clear. We hypothesise that the degradation in Turkish could perhaps be attributed to the fact that the language is highly agglutinative in nature. Agglutination is a linguistic process of derivational morphology in which complex words are formed by stringing together morphemes without changing them in spelling or phonetics (Wikipedia contributors, 2021a). While Japanese and Hindi do show partial agglutination, the morphemes/words are much more complex in Turkish. This hypothesis needs to be further investigated by checking for zero-shot performance on other agglutinative languages such as Hungarian, languages of the Dravidian family etc and the investigation forms the part of our future work.

5 Conclusion and Future Directions

In this work, we have investigated the effectiveness of multi-task pre-finetuning for cross lingual zero-shot transfer. Our method takes a multilingual pre-trained model and further trains it on auxiliary supervised tasks. The pre-finetuned model is then finetuned on a task specific language and tested directly on other languages in a zero-shot setting. We test our method for the tasks of Intent Classification (IC) and Named Entity Recognition (NER). The results indicate that the method indeed improves the performance for the IC task. This improvement is seen the most in early steps of finetuning and our method allows the training to converge faster. However, we see that the pre-finetuning does not improve results for NER. Further, we see that both IC and NER results degrade in Turkish.

Our furture directions include scaling our method to cover large number of auxiliary tasks for pre-finetuning. While our current method used 8 auxuliary tasks, we aim to scale this to beyond 50. Large scale multi-task learning has been shown to be effective in a monolingual setting (Aghajanyan et al., 2021) and we would like to explore this phenomenon in a multilingual setting. We will also explore the role of language families and its interaction with multi-task learning to test the hypothesis of poor performance in agglutinative languages (such as Turkish).

The pratical application of zero-shot learning provides a data-efficient method to expand the language capability of machine learning based techniques. The results from our technique show that such zero-shot performance can be further improved and also provide impetus for further research.

References

A. Aghajanyan, A. Gupta, A. Shrivastava, X. Chen, L. Zettlemoyer, and S. Gupta. Muppet: Massive multi-task representations with pre-finetuning. *CoRR*, abs/2101.11038, 2021. URL `https://arxiv.org/abs/2101.11038`.

Q. Chen, Z. Zhuo, and W. Wang. BERT for joint intent classification and slot filling. *CoRR*, abs/1902.10909, 2019. URL `http://arxiv.org/abs/1902.10909`.

A. Conneau, R. Rinott, G. Lample, A. Williams, S. Bowman, H. Schwenk, and V. Stoyanov. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1269. URL `https://www.aclweb.org/anthology/D18-1269`.

A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov. Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116, 2019. URL http://arxiv.org/abs/1911.02116.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 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, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Multilingual BERT. https://github.com/google-research/bert/blob/master/multilingual.md, 2021. [Online; accessed 27-May-2021].

S. Khanuja, D. Bansal, S. Mehtani, S. Khosla, A. Dey, B. Gopalan, D. K. Margam, P. Aggarwal, R. T. Nagipogu, S. Dave, S. Gupta, S. C. B. Gali, V. Subramanian, and P. Talukdar. Muril: Multilingual representations for indian languages. *CoRR*, abs/2103.10730, 2021. URL https://arxiv.org/abs/2103.10730.

J. Libovický, R. Rosa, and A. Fraser. On the language neutrality of pre-trained multilingual representations. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1663–1674, Online, Nov. 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.150. URL https://www.aclweb.org/anthology/2020.findings-emnlp.150.

J. Libovický, R. Rosa, and A. Fraser. On the language neutrality of pre-trained multilingual representations. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1663–1674, Online, Nov. 2020b. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.150. URL https://www.aclweb.org/anthology/2020.findings-emnlp.150.

X. Liu, P. He, W. Chen, and J. Gao. Multi-task deep neural networks for natural language understanding. *CoRR*, abs/1901.11504, 2019. URL http://arxiv.org/abs/1901.11504.

X. Liu, Y. Wang, J. Ji, H. Cheng, X. Zhu, E. Awa, P. He, W. Chen, H. Poon, G. Cao, and J. Gao. The microsoft toolkit of multi-task deep neural networks for natural language understanding. *CoRR*, abs/2002.07972, 2020. URL https://arxiv.org/abs/2002.07972.

A. Radford and I. Sutskever. Improving language understanding by generative pre-training. 2018. URL https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf.

E. Schlinger. How multilingual is Multilingual BERT ? pages 4996–5001, 2019a.

E. Schlinger. How multilingual is Multilingual BERT ? pages 4996–5001, 2019b.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *CoRR*, abs/1804.07461, 2018. URL http://arxiv.org/abs/1804.07461.

Wikipedia contributors. Agglutination — Wikipedia, the free encyclopedia, 2021a. URL https://en.wikipedia.org/w/index.php?title=Agglutination&oldid=1029218007. [Online; accessed 28-June-2021].

Wikipedia contributors. Languages of india — Wikipedia, the free encyclopedia, 2021b. URL https://en.wikipedia.org/w/index.php?title=Languages_of_India&oldid=1030136775. [Online; accessed 26-June-2021].

S. Wu and M. Dredze. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. 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 833–844, Hong Kong, China, Nov. 2019a. Association for Computational Linguistics. doi: 10.18653/v1/D19-1077. URL https://www.aclweb.org/anthology/D19-1077.

S. Wu and M. Dredze. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. 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 833–844, Hong Kong, China, Nov. 2019b. Association for Computational Linguistics. doi: 10.18653/v1/D19-1077. URL https://www.aclweb.org/anthology/D19-1077.

W. Xu, B. Haider, and S. Mansour. End-to-end slot alignment and recognition for cross-lingual nlu. *arXiv preprint arXiv:2004.14353*, 2020.

Sentiment Analysis For Bengali Using Transformer Based Models

Anirban Bhowmick
Universität Hamburg
anirbanbhowmick88@gmail.com

Abhik Jana
Universität Hamburg
abhikjana1@gmail.com

Abstract

Sentiment analysis is one of the key Natural Language Processing (NLP) tasks that has been attempted by researchers extensively for resource-rich languages like English. But for low resource languages like Bengali very few attempts have been made due to various reasons including lack of corpora to train machine learning models or lack of gold standard datasets for evaluation. However, with the emergence of transformer models pre-trained in several languages, researchers are showing interest to investigate the applicability of these models in several NLP tasks, especially for low resource languages. In this paper, we investigate the usefulness of two pre-trained transformers models namely multilingual BERT and XLM-RoBERTa (with fine-tuning) for sentiment analysis for the Bengali Language. We use three datasets for the Bengali language for evaluation and produce promising performance, even reaching a maximum of 95% accuracy for a two-class sentiment classification task. We believe, this work can serve as a good benchmark as far as sentiment analysis for the Bengali language is concerned.

1 Introduction

In this era of the World Wide Web, sharing of information, knowledge, opinion, etc. has been increased by a huge margin since the last decade. Internet users are coming forward to review stuff like books, movies, videos, e-commerce products, etc., and are sharing their experiences which in turn help the next in line users to get feedback upfront. This genre of texts brings in the essence of sentiment analysis task which helps in polarity classification, i.e. determining whether a given text expresses positive, negative, or neutral sentiment. Sentiment analysis of texts can be useful for different applications, like detecting cyberbullying (Saravanaraj et al., 2016), hate speech detection (von Boguszewski et al., 2021; Mathew et al., 2021), e-commerce recommendation system (Hwangbo et al., 2018), etc. There has been a substantial amount of work done by the researchers to tackle sentiment analysis for resource-rich languages like English (Pak and Paroubek, 2010; Feldman, 2013), but for low resource languages, such attempts are scarce (Islam et al., 2020; Sazzed, 2020; Siripragrada et al., 2020). In recent times, for low resource languages like Hindi, Telegu, Bengali, Assamese, Manipuri, Indonesian, etc. (Akhtar et al., 2016; Mukku and Mamidi, 2017; Sazzed, 2020; Le et al., 2016; Kumar and Albuquerque, 2021; Meetei et al., 2021; Das and Singh, 2021; Singh et al., 2021; Kumari et al., 2021) and even for English-Hindi, English- Bengali code-mixed languages (Jamatia et al., 2020), researchers have come up with a solution for sentiment analysis tasks. In another work, R et al. (2012) performed cross-lingual sentiment analysis task where the opinion polarity of a text in a language is predicted using classifier trained in another language. The authors report results on two widely spoken Indian languages, Hindi and Marathi. Gupta et al. (2021) used an LSTM-RNN based approach to determine the sentiment of Hindi tweets and also compared their approach with CNN, machine learning, and Lexicon based approaches. Gupta et al. (2021) uses Hindi SentiWordNet (HSWN) proposed by Joshi et al. (2010) as a lexicon generating tool for hindi text. So Hindi being a major Indian language has been explored whereas more insights are still needed in Bengali. In one of the very recent works, Islam et al. (2020) prepare a two-class and a three-class sentiment analysis dataset in Bengali and report performances of multilingual BERT (Devlin et al., 2019) which is impressive. Moving forward in a similar direction, in this paper we apply two pre-trained transformers models namely multilingual

BERT and XLM-Roberta (Conneau et al., 2020) after fine-tuning and conducting the analysis. In addition to the datasets proposed by Islam et al. (2020), we use two other datasets proposed by Sazzed (2020) and Hossain et al. (2021) for our study. We observe that, by applying fine-tuned multilingual BERT and XLM-RoBERTa (for convenience we will refer XLM-Roberta as XLM-R in our paper), we achieve an accuracy of 63%-94% and 68%-95%, while evaluating against these three target datasets leading to state-of-the-art performances. To the best of our knowledge, this is the first attempt at such a comprehensive study for the Bengali language where pre-trained transformer models' applicability (with fine-tuning) has been investigated for sentiment analysis tasks and evaluated against three datasets. All the codes and datasets are made publicly available[1].

2 Dataset

For this study, we use three datasets. The details of these datasets are described below.

Prothom Alo: This is the first dataset[2] used in this study which is a publicly available dataset created from user comments on 10 popular news topics from an online Bengali news portal, Prothom Alo[3]. This dataset is introduced by Islam et al. (2020), for convenience, we refer to this dataset as 'Prothom Alo'. The authors scrape user comments from news threads and clean to obtain a total of 17,852 user comments. Each of the comments is tagged by Bengali domain experts into one of the following three classes: positive, negative, and neutral. The authors prepare a variant of this dataset as well which has only two classes by removing the neutral class entries. This step results in a dataset for two-class classification with 13,120 entries.

YouTube-B: This is a collection of reviews manually annotated from YouTube Bengali drama[4] consisting of 8500 positive reviews and 3307 negative reviews and is introduced by Sazzed (2020). This dataset is a two-class dataset having only positive and negative as labels. We refer to this dataset as 'YouTube-B' for the rest of the paper. 'B' stands for Bengali language.

[1]https://github.com/Anirbanbhk88/BengaliSentimentWithTransformers

[2]https://github.com/KhondokerIslam/Bengali_Sentiment

[3]https://www.prothomalo.com/

[4]https://data.mendeley.com/datasets/p6zc7krs37/4

#(Classes)	Dataset	Neu	Pos	Neg
Three	Prothom Alo	4732	4769	8351
Two	Prothom Alo	-	4769	8351
	YouTube-B	-	8500	3307
	Book-B	-	982	1018

Table 1: Class Distribution of Bengali sentiment analysis datasets. 'Neu', 'Pos' and 'Neg' represent neutral, positive, and negative classes, respectively.

Book-B: This is the third dataset introduced by Hossain et al. (2021). It is a collection of Bengali book reviews collected from web resources such as blogs, Facebook, and e-commerce sites. This dataset is also a two-class dataset (having positive and negative classes) with 2000 entries of book reviews. We refer to this dataset as 'Book-B' for the rest of the paper.

The details of these three datasets are shown in Table 1.

3 Proposed Approach

In this study, we consider the state-of-the-art multilingual BERT model (Devlin et al., 2019) and XLM-RoBERTa model (Conneau et al., 2020) for the Bengali sentiment analysis task. First, we use them separately in one of the recent architecture proposed by Islam et al. (2020), where authors use Long Short Term Memory (LSTM) (Cho et al., 2014), Convolutional Neural Network (CNN) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Cho et al., 2014)) on top of the transformer model. Next, we fine-tune the pre-trained BERT and XLM-RoBERTa model using each of the three datasets separately (as depicted in Figure 1) and analyze the performances. In both the direction of exploration, BERT and XLM-RoBERTa are the core transformers for our analysis. Hence, a summary of both the BERT model and the XLM-RoBERTa (XLM-R) model is described below.

Description of models: BERT and XLM-R are unsupervised language models pre-trained on a large corpus. They are transformer-based models which have encoder-decoder architecture and use attention mechanisms to generate a contextualized representation of words. BERT uses a multi-layer bidirectional transformer encoder. Its self-attention layer performs self-attention in both directions. There are several variants of BERT. For example, *bert-base* has 12 transformers layers, 110M total parameters while *bert-large* has 24 transformers layers, 340M total parameters. They are useful to solve the long-range dependencies which is a key problem faced by sequence to sequence models like Recurrent Neural Networks(RNN). For our proposed approach we use *bert-base-multilingual-cased*, which is a *bert-base* model checkpoint trained on multilingual corpus.

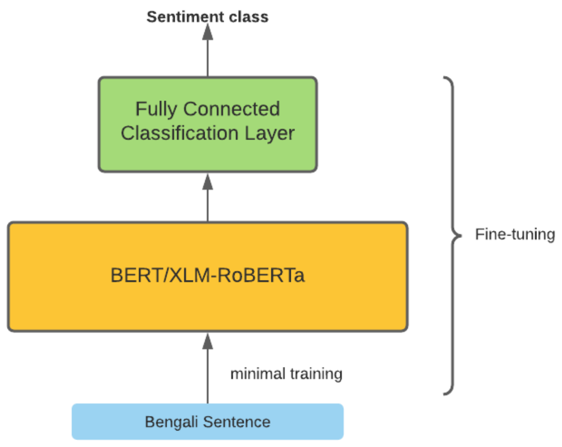


Figure 1: A snapshot of the architecture we fine-tuned for the sentiment analysis task.

Dataset (#Classes)	**Layer on top of BERT**	**Validation Accuracy**
Prothom Alo (3)	LSTM	0.58
	CNN	**0.59**
	GRU	0.57
Prothom Alo (2)	LSTM	0.63
	CNN	**0.74**
	GRU	**0.74**
YouTube-B (2)	LSTM	0.85
	CNN	**0.92**
	GRU	0.91
Book-B (2)	LSTM	0.49
	CNN	**0.91**
	GRU	0.86

Table 2: Accuracy of the framework proposed by (Islam et al., 2020) where LSTM, CNN and GRU are used on top of multilingual BERT.

XLM indicates a cross-lingual language model. XLM-RoBERTa (XLM-R) is a pre-trained multilingual model that is considered to be superior over multilingual BERT when evaluated against various NLP tasks. One probable reason could be that XLM-R is trained using a much bigger corpus. XLM-R is also trained in approximately 100 languages. Similarly incase of XLM-R, in our approach we use *xlm-roberta-large* checkpont of pre-trained XLM-R model. We opt for the idea of fine-tuning BERT and XLM-R models especially for low resource language like Bengali, as fine-tuning can be done with a small amount of training data, and the training process is also less time consuming since we are not training all the layers from scratch. Note that, all these transformer-based models used in our study are adopted from HuggingFace.[5].

Dataset (#Classes)	**Layer on top of XLM-R**	**Validation Accuracy**
Prothom Alo (3)	LSTM	**0.65**
	CNN	0.37
	GRU	0.63
Prothom Alo (2)	LSTM	0.64
	CNN	0.77
	GRU	**0.79**
YouTube-B (2)	LSTM	0.85
	CNN	**0.90**
	GRU	**0.90**
Book-B (2)	LSTM	0.60
	CNN	**0.88**
	GRU	0.84

Table 3: Accuracy of the variant of the framework proposed by (Islam et al., 2020) where LSTM, CNN, and GRU are used on top of XLM-R.

Dataset (#Classes)	**Models**	**Validation Accuracy**	**Test Accuracy**
Prothom Alo (3)	BERT	0.63	0.49
	XLM-R	**0.68**	**0.53**
Prothom Alo (2)	BERT	0.77	0.69
	XLM-R	**0.81**	**0.73**
YouTube-B (2)	BERT	0.94	0.95
	XLM-R	**0.95**	**0.97**
Book-B (2)	BERT	**0.91**	**0.91**
	XLM-R	**0.91**	0.87

Table 4: Validation (Val) and Test accuracy(Acc) of fine-tuned BERT and XLM-R models all the three datasets respectively. XLM-R here represents XLM-RoBERTa.

4 Experimental Setup

For the series of experiments performed, we first adopt the model from the work by Islam et al. (2020) and use it as a baseline model. This benchmark model consists of a multilingual BERT (*bertbase-multilingual-cased*) pre-trained on multiple languages. Three different deep neural network layers: GRU, LSTM, CNN are used as an extra layer on top of BERT separately to produce three separate architectures. We use their code repository and train the baseline models using the same set of hyper-parameters and attempt to replicate the results. Next, we replace BERT with XLM-R in the same architecture. As XLM-R models we use *xlm-roberta-large*. For this set of exper-

[5]https://huggingface.co/transformers

Samples	Model	#(Classes)	Target	Prediction
প্রথম আলোর এই ডিপারমেন্ট খুব কাঁচা রিপোর্টের দিক দিয়ে । This department of Prothom Alo is very raw in terms of reporting.	BERT-Fine	3	Neg	Neg
খেলাধুলায় ভ্রাতৃত্ববোধ থাকা প্রয়োজন । রেষারেষি নয় । There needs to be a sense of brotherhood in sports. Not a rivalry.	BERT-Fine	3	Pos	Pos
সবার সাথেই সেম অবস্থা , বুঝতে হবে আমাদের ন্যাচার The situation is same with everyone, we have to understand nature.	BERT-Fine	3	Neu	Pos
অসাধারন নাটক এমন টা হয়েছিলো আমার সাথে মত । Such an extraordinary drama. The same happened to me.	BERT-Fine	2	Pos	Pos
প্রথম আলোর এই ডিপারমেন্ট খুব কাঁচা রিপোর্টের দিক দিয়ে । This department of Prothom Alo is very raw in terms of reporting.	BERT-Fine	2	Neg	Pos
প্রথম আলোর এই ডিপারমেন্ট খুব কাঁচা রিপোর্টের দিক দিয়ে । This department of Prothom Alo is very raw in terms of reporting.	XLM-R-Fine	3	Neg	Neg
আমরা এমন রেসারেসি চাই না , সবার সাথে বন্ধুত্ব পূর্ণ সম্পর্ক চাই । We do not want such a race, we want a friendly relationship with everyone.	XLM-R-Fine	3	Pos	Pos
ঘুমন্ত ব্যক্তিকে জাগানো যায় কিন্তু জাগ্রতকে নয় The sleeping person can be awakened but not watchful.	XLM-R-Fine	3	Pos	Neu
শুধুই ভালোবাসা নিশ ভাই Take Only love brother.	XLM-R-Fine	2	Pos	Pos
কুরুচিপূরণ শব্দে ভরতি বই A book full of ugly words.	XLM-R-Fine	2	Neg	Pos

Table 5: Sample predictions for fine-tuned BERT and XLM-RoBERTa extracted from different datasets. 'Target' column represents gold standard class as per dataset and 'Prediction' column represents predicted class by our models. 'Neu', 'Pos' and 'Neg' represents the neutral, positive and negative class.

iments, we use a learning rate of $5e^{-}04$.

We also perform another set of experiments where we use pre-trained BERT(*bert-base-multilingual-cased*) and XLM-R(*xlm-roberta-large*) and fine-tune them. For all the fine-tuning experiments a batch size of 16, a learning rate of $2e^{-}05$, and a categorical cross-entropy loss function are used. We use Adam optimizer (Kingma and Ba, 2015) for all the experiments. More details of hyper-parameters used are mentioned in Table 1 of supplementary material.

5 Results and Discussion

Even though our primary aim is to investigate the applicability of fine-tuned multilingual BERT and XLM-RoBERTa for Bengali sentiment analysis task, we start our experiment with one of the most recent baseline models proposed by Islam et al. (2020). As Islam et al. (2020) perform all their evaluation on their proposed dataset, Prothom Alo, we first reproduce their result on the same dataset which is presented in the upper half of Table 2. In addition to that, we also evaluate their models on Youtube-B and Book-B datasets as well which are presented in the bottom half of Table 2. We observe BERT with CNN produces an accuracy as high as 0.92 and 0.91 for Youtube-B and Book-B, respectively. Note that, in the study done by Islam et al. (2020), authors report accuracy for the validation set. Therefore, to make a fair comparison we also report the same. Next, we investigate further by using the same model architectures but instead of using multilingual BERT, replacing it with XLM-RoBERTa (XLM-R). The performances of this modified architecture over all three datasets are presented in Table 3. The result shows, that replacing multilingual BERT with XLM-R improves the performance for Prothom Alo dataset (for both three class and two-class classification tasks) by a maximum of 7%. On the other hand, for Youtube-B and Book-B datasets the performance marginally reduces.

Such inconsistencies in performances over different models lead to our next step which deals with fine-tuning multilingual BERT and RoBERTa using three datasets. Note that, in this approach, we do not use any of the LSTM, CNN, and GRU layers on top of the transformer layers as it was done by Islam et al. (2020) as we attempt to show that rather than implementing custom and complex architectures working well on a specific task, simply fine-tuning a transformer is an easier, better alternative. The results of this approach are presented in Table 4. We see that, for 'Prothom Alo' (both two and three classification tasks) fine-tuned XLM-R beats all the previous approaches discussed so far by a significant margin and achieves validation accuracy of 0.68 for the three-class classification task and 0.81 for the two-class classification task. For 'Youtube-B' fine-tuned XLM-R produces an accuracy of 0.95 whereas for 'Book-B' it produces an accuracy of 0.91 which looks promising.

Note that, to have a fair comparison with the most recent baseline models proposed by (Islam et al., 2020), we report validation accuracy following their performance measures and we see fine-tuned XLM-R outperforms this baseline by a significant margin for the 'Prothom Alo' dataset. In addition, we also report test accuracy in the last column of Table 4. Fine-tuned BERT/XLM-R produces substantially improved performances over closest baselines which will serve as new state-of-the-art performance for these three datasets.

We further investigate a few predicted samples from different datasets to check for the cases that were predicted wrongly by fine-tuned BERT or XLM-R . Few correctly predicted and wrongly predicted samples are presented in Table 5. Even though the overall fine-tuned BERT/XLM-R model performs well, there are certain cases where these models get confused and predict wrongly. In most such cases, the Bengali sentence either contains an ambiguous word or it contains two words from different polarity or it contains some sort of philosophy the meaning of which depends on human interpretation. Taking care of these such cases could be immediate future work.

6 Conclusion

In this paper, we conduct an experimental study showing the applicability of multilingual BERT and XLM-R (with fine-tuning) for the Bengali sentiment analysis task. We use three datasets to evaluate the models and obtain promising performances for all three datasets. The immediate future step would be investigating the erroneously classified cases and trying to find the reason behind such errors and mitigate it. Broadly, we plan to investigate sentiment analysis for other low-resource languages like Tamil, Oriya, Gujrati, etc and attempt to propose variants of transformer based models.

References

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, Osaka, Japan. The COLING 2016 Organizing Committee.

Niklas von Boguszewski, Sana Moin, Anirban Bhowmick, Seid Muhie Yimam, and Chris Biemann. 2021. How hateful are movies? a study and prediction on movie subtitles. In *Proceedings of the 17th Conference on Natural Language Processing (KONVENS 2021)*, pages 37–48, Düsseldorf, Germany. KONVENS 2021 Organizers.

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. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. 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.

Ringki Das and Thoudam Doren Singh. 2021. A step towards sentiment analysis of assamese news articles using lexical features. In *Proceedings of the International Conference on Computing and Communication Systems: I3CS 2020, NEHU, Shillong, India*, volume 170, page 15. Springer.

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, MN, USA. Association for Computational Linguistics.

Ronen Feldman. 2013. Techniques and applications for sentiment analysis. *Commun. ACM*, 56(4):82–89.

Vedika Gupta, Nikita Jain, Shubham Shubham, Agam Madan, Ankit Chaudhary, and Qin Xin. 2021. Toward integrated cnn-based sentiment analysis of tweets for scarce-resource language—hindi. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 20(5).

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8):1735—1780.

Eftekhar Hossain, Omar Sharif, and Mohammed Moshiul Hoque. 2021. Sentiment polarity detection on bengali book reviews using multinomial naive bayes. In *Progress in Advanced Computing and Intelligent Engineering*, pages 281–292. Springer.

Hyunwoo Hwangbo, Yang Sok Kim, and Kyung Jin Cha. 2018. Recommendation system development for fashion retail e-commerce. *Electronic Commerce Research and Applications*, 28:94–101.

Khondoker Ittehadul Islam, Md Saiful Islam, and Md Ruhul Amin. 2020. Sentiment analysis in bengali via transfer learning using multi-lingual bert. In *2020 23rd International Conference on Computer and Information Technology (ICCIT)*, pages 1–5. IEEE.

Anupam Jamatia, Steve Durairaj Swamy, Björn Gambäck, Amitava Das, and Swapan Debbarma. 2020. Deep learning based sentiment analysis in a code-mixed english-hindi and english-bengali social media corpus. *International Journal on Artificial Intelligence Tools*, 29(05):2050014.

Aditya Joshi, AR Balamurali, Pushpak Bhattacharyya, et al. 2010. A fall-back strategy for sentiment analysis in hindi: a case study. *Proceedings of the 8th ICON*.

Diederik P. Kingma and Jimmy Ba. 2015. ADAM: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015*, pages 1–15, San Diego, CA, USA.

Akshi Kumar and Victor Hugo C Albuquerque. 2021. Sentiment analysis using xlm-r transformer and zero-shot transfer learning on resource-poor indian language. *Transactions on Asian and Low-Resource Language Information Processing*, 20(5):1–13.

Divya Kumari, Asif Ekbal, Rejwanul Haque, Pushpak Bhattacharyya, and Andy Way. 2021. Reinforced nmt for sentiment and content preservation in low-resource scenario. *Transactions on Asian and Low-Resource Language Information Processing*, 20(4):1–27.

Tuan Anh Le, David Moeljadi, Yasuhide Miura, and Tomoko Ohkuma. 2016. Sentiment analysis for low resource languages: A study on informal indonesian tweets. In *Proceedings of the 12th Workshop on Asian Language Resources (ALR12)*, pages 123–131.

Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2021. HateXplain: A Benchmark Dataset for Explainable Hate Speech Detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(17):14867–14875.

Loitongbam Sanayai Meetei, Thoudam Doren Singh, Samir Kumar Borgohain, and Sivaji Bandyopadhyay. 2021. Low resource language specific preprocessing and features for sentiment analysis task. *Language Resources and Evaluation*, pages 1–23.

Sandeep Sricharan Mukku and Radhika Mamidi. 2017. ACTSA: Annotated corpus for Telugu sentiment analysis. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*, pages 54–58, Copenhagen, Denmark. Association for Computational Linguistics.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*, volume 10, pages 1320–1326.

Balamurali R, Aditya Joshi, and Pushpak Bhattacharyya. 2012. Cross-lingual sentiment analysis for indian languages using linked wordnets. pages 73–82.

A Saravanaraj, JI Sheeba, and S Pradeep Devaneyan. 2016. Automatic detection of cyberbullying from twitter. *International Journal of Computer Science and Information Technology & Security (IJCSITS)*.

Salim Sazzed. 2020. Cross-lingual sentiment classification in low-resource Bengali language. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 50–60, Online. Association for Computational Linguistics.

Thoudam Doren Singh, Telem Joyson Singh, Mirinso Shadang, and Surmila Thokchom. 2021. Review comments of manipuri online video: Good, bad or ugly. In *Proceedings of the International Conference on Computing and Communication Systems: I3CS 2020, NEHU, Shillong, India*, volume 170, page 45. Springer.

Shashank Siripragrada, 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.

IndicFed: A Federated Approach for Sentiment Analysis in Indic Languages

Jash Mehta*, Deep Gandhi*, Naitik Rathod, Sudhir Bagul[‡]
Dwarkadas J. Sanghvi College of Engineering
Mumbai, India
{jashmehta3300,thisisdeepgandhi,naitikrathod18}@gmail.com
[‡]sudhir.bagul@djsce.ac.in

Abstract

The task of sentiment analysis has been extensively studied in high-resource languages. Even though sentiment analysis is studied for some resource-constrained languages, the corpora and the datasets available in other low resource languages are scarce and fragmented. This prevents further research of resource-constrained languages and also inhibits model performance for these languages. Privacy concerns may also be raised while aggregating some datasets for training central models. Our work tries to steer the research of sentiment analysis for resource-constrained languages in the direction of Federated Learning. We conduct various experiments to compare server based and federated approaches for 4 Indic Languages - Marathi, Hindi, Bengali, and Telugu. Specifically, we show that a privacy preserving approach, Federated Learning surpasses traditional server trained LSTM model and exhibits comparable performance to other servers-side transformer models.

1 Introduction

With the proliferation of opinionated user data on social media platforms (Murphy et al., 2014), capturing user emotions could help in decision making and determining public opinion on cultural, social, and political agendas (Zhao et al., 2016; Liu, 2012). This has prompted research into sentiment analysis and opinion mining for English (e.g. Thelwall et al., 2010, 2012; Li and Lu, 2017; Hussein, 2018; Li and Lu, 2019; Li et al., 2019; Hoang et al., 2019; Ruz et al., 2020; Chen et al., 2021), which is aided by the availability of large-scale, centralized training datasets. However, there is a pressing need to work on NLP beyond resource-rich languages due to cultural, linguistic, and societal factors (Ruder, 2020).

* indicates equal contribution

Sentiment Analysis in low-resource Indic languages has posed a challenge to the research community due to the absence of large-scale centralized datasets. Moreover, to due to increasing concerns and regulations about data privacy (e.g. GDPR (Regulation, 2016)), emerging data has been much more fragmented. It resides in decentralized private silos across different client devices. To abide by such regulations and respect the privacy of users, we must assume that these private data silos can not be shared either with other clients or with the centralized server. Hence, it is exigent to tackle these challenges and study the problem of sentiment analysis in a much more realistic setting - i.e., training models on distributed data silos across different clients to maintain data privacy.

Federated Learning (FL) (McMahan et al., 2017), is a distributed learning paradigm which aims to enable individual clients to train their models collaboratively while keeping their local data private. Instead of accumulating data on a centralized server for training the model, each client sends its model parameters to the server, which updates and sends back the global model to all clients in each round. Since the raw data always remains on the client device and is never shared, FL offers promising solution to the above challenges, particularly in resource poor languages where collection of large-scale training data is difficult.

Previous works for sentiment analysis have relied on traditional server-based architectures and have been centered around resource rich languages. However, such models risk leakage of highly sensitive user-generated data. Thus, we propose a privacy-preserving approach, Federated Learning for sentiment analysis in 4 Indic languages. To the best of our knowledge, our work is the first effort towards Federated learning on Marathi, Bengali, and Telugu datasets; and also towards sentiment analysis in Hindi.

Language	Dataset	Reference	# Classes	# Examples			
				Train	Test	Dev	Total
Marathi (mr)	L3CubeMahaSent	Kulkarni et al. (2021)	3	12,114	2,250	1,500	15,864
Telugu (te)	ACTSA	Mukku and Mamidi (2017)	3	3,784	812	812	5,408
Bengali (bn)	ABSA Cricket	Rahman et al. (2018)	3	2,085	380	372	2,837
	ABSA Restaurant	Rahman et al. (2018)	3	1,365	219	224	1,808
	YouTube Comments	Tripto and Ali (2018)	3	1,957	420	419	2,796
	SAIL	Patra et al. (2015)	3	697	204	98	999
	BengFastText	Karim et al. (2020)	2	5,510	1,532	1,378	8,420
	Combined	Hasan et al. (2020)	3	9,901	2,755	2,491	15,147
Hindi (hi)	Hindi Sentiment Analysis	Sinha (2019) [1]	3	6,353	1,362	1,362	9,077

Table 1: Summary of different datasets

We use of publicly available datasets for sentiment analysis in Marathi (Kulkarni et al., 2021), Telugu (Mukku and Mamidi, 2017), Bengali (Hasan et al., 2020), and Hindi [1]. We examine how Federated LSTM model performs, in comparison to 4 server-side centralized models: bi-directional LSTM (Hochreiter and Schmidhuber, 1997), IndicBert (Kakwani et al., 2020b), mBERT (Devlin et al., 2019), and XLM-R (Conneau et al., 2020). We find that the federated learning architecture outperforms the centralized server-side LSTM model and shows comparable performance to the centralized transformer models for all 4 languages under consideration.

The remainder of the paper is organized into prior work (§2), a brief description of the datasets we use (§3), a description of the experimental setup (§4), an in-depth analysis our experiments (§5), and finally a conclusion (§6).

2 Prior Work

Federated Learning: Federated Learning (McMahan et al., 2017) is used for building PPML (Privacy Preserving Machine Learning) models. As proposed by Hard et al. (2018), Federated Learning is useful for preventing bottlenecks when the data is trained on central servers. Some major work is being done for the English language at the intersection of Federated Learning and Natural Language Processing which was also observed in Lin et al. (2021). However, even though benchmarks were established for English language, the task of using it on Indic resource-constrained languages remains relatively unexplored. Recently, Singh et al. (2021) showed that Federated Learning surpasses the baselines for complaint identification in some Indic languages. It is evident from these approaches that using this technique helps achieve better results. Therefore, we consider a total of 4 Indic languages (Hindi, Marathi, Telugu, Bengali) in this paper to conduct Federated Learning.

Sentiment Analysis in Indic Languages: Joshi et al. (2010) talks about various approaches for sentiment analysis in Hindi, and resorts to translating the data to English for sentiment identification due to the issue of constrained resources for Hindi. However, even though Hindi cannot be considered a very resource-constrained language now due to the development of various corpora such as (e.g. Kunchukuttan et al., 2018; Khandelwal et al., 2018; Bafna and Saini, 2021), but a lot of other Indic languages are still resource constrained and corpora for the same are very limited. The datasets for such Indic languages are spread out as observed for Bengali in Table 1. Many approaches have been adopted server trained models for sentiment analysis of the languages being considered here (e.g. Salehin et al., 2020; Regatte et al., 2020; Kulkarni et al., 2021; Jain et al., 2020; Hasan et al., 2020; Kakwani et al., 2020a).

3 Datasets

We use publicly available datasets for sentiment analysis in low-resource Indian languages. Table 1 gives the details of all datasets used in our work. We combine all the Bengali datasets and train our models on the combined dataset.

In case of ACTSA Dataset (Mukku and Mamidi, 2017) and Hindi Sentiment Analysis Dataset [1], we make a stratified split of 70:15:15 to divide the data into train, test, and development sets. For other datasets, we use the original splits which are provided.

[1]`https://github.com/sid573/Hindi_Sentiment_Analysis`

	mBERT			**XLM-R**			**IndicBERT**		
	Acc.	AUC	F1	Acc.	AUC	F1	Acc.	AUC	F1
te	45.81 ± 0.00	49.46 ± 1.66	28.79 ± 0.00	48.45 ± 5.89	54.98 ± 10.34	34.81 ± 13.46	$\mathbf{61.85 \pm 0.53}$	$\mathbf{76.61 \pm 0.69}$	$\mathbf{61.74 \pm 0.62}$
hi	75.57 ± 3.93	87.03 ± 3.83	74.55 ± 5.02	88.39 ± 0.39	94.93 ± 1.38	87.40 ± 2.49	$\mathbf{88.58 \pm 1.61}$	$\mathbf{95.73 \pm 0.72}$	$\mathbf{88.58 \pm 1.62}$
bn	77.54 ± 0.64	80.66 ± 1.01	76.72 ± 0.67	78.30 ± 6.87	81.24 ± 7.58	77.00 ± 8.28	$\mathbf{80.87 \pm 0.26}$	$\mathbf{86.27 \pm 0.76}$	$\mathbf{80.35 \pm 0.23}$
mr	69.98 ± 0.57	83.58 ± 0.43	69.97 ± 0.55	82.47 ± 0.46	92.02 ± 0.62	82.42 ± 0.47	$\mathbf{83.36 \pm 0.36}$	$\mathbf{93.51 \pm 0.36}$	$\mathbf{83.33 \pm 0.35}$

Table 2: Performance of centrally trained models on 5 different seeds. IndicBERT performs better than the others for all languages.

3.1 Pre-processing

To pre-process the data, we lower-case all text and remove numbers, punctuation, and URLS. Since some of the datasets are taken from Twitter, we also remove Twitter specific things like hashtags, @-mentions, and the retweet marker: "RT:".

4 Experiments

All the experiments are conducted on Google Colab using a NVIDIA Tesla P100 GPU (16 GB) with 26 GB RAM. The metrics used to compare the results are weighted AUC, weighted F1 and the accuracy score for every model in every variation.

4.1 Central Training

In order to compare results to Federated Learning, we use 4 different models: mBERT (Devlin et al., 2019), XLM-R (Conneau et al., 2020), IndicBERT (Kakwani et al., 2020b), and Bi-LSTM (Hochreiter and Schmidhuber, 1997). We report the mean and standard deviation after training on 5 random seeds. All of these models except the Bi-LSTM architecture are pretrained on the languages being considered in this paper.

Every pretrained model is trained for 25 epochs and the Bi-LSTM model is run for 500 epochs with early stopping. The default learning rate ($lr = 4e-5$) is used for the pretrained models and for the Bi-LSTM model, the learning rate of 0.01 is set.

4.2 Federated Learning

We use the FedProx algorithm (Li et al., 2018) because it works better in non-iid data where the distribution varies rapidly within the dataset. We conduct various experiments under synthetic-iid (independent and identical distribution) (Li et al., 2018) and non-iid settings. Since, some of these Indic languages cannot be tokenized using general tokenizers such as Spacy[2], we use language specific tokenizers provided by iNLTK (Arora, 2020).

To make the computation cheaper on resource-constrained edge-devices, by the distributed training process, we train it on a basic Bi-LSTM model (Hochreiter and Schmidhuber, 1997) with 2 hidden layers and dropout (Srivastava et al., 2014) set to 0.5. Different client fractions are used to observe the variation of results in the Federated setting too. The client fractions of 10%, 30% and 50% are considered and these clients are always picked randomly for every round.

All the models are run for 500 rounds with early stopping applied on the average training loss. The learning rate is set to 0.01 and the proximal term is set to 0.01 as default (Li et al., 2018).

5 Results

Telugu (te): From Table 4, it can be observed that for the Telugu language, the Federated training process performs much better than the centrally trained LSTM model. The best model chosen for the Federated setting is with $c = 30\%$. Even though the Federated training is trained on 30% of the data every round, the results are better than the central model. Looking at the other models trained for the Telugu language, we also observe from Table 2 that the best performing model IndicBERT (Kakwani et al., 2020b) has comparable results to the Federated LSTM model[3].

Hindi (hi): From Table 4, we find that the federated model performs better on all the metrics than the server-based LSTM model. Even though we achieve better F1 score for $c = 30\%$ (Table 3), we consider the model with $c = 10\%$ as the high performing model for federated learning because of the AUC score and the accuracy. It must be noted that the scores are on the lower side for models trained on non-iid setting in federated learning because every client cluster is intentionally biased to represent one single class unlike the synthetic-iid method.

[2]`https://spacy.io/`

[3]IndicBERT was pretrained on 674M tokens of Telugu. (Kakwani et al., 2020b)

	c = 10%						**c = 30%**						**c = 50%**					
	IID			non-IID			IID			non-IID			IID			non-IID		
	Acc	AUC	F1	Acc	AUC	F1	Acc	AUC	F1	Acc	AUC	F1	Acc	AUC	F1	Acc	AUC	F1
te	47.62	**76.09**	**55.75**	38.10	64.42	37.74	**66.67**	74.32	54.38	42.86	66.88	25.54	61.91	71.16	49.90	38.10	68.33	32.32
hi	**81.82**	**99.79**	84.18	45.46	92.18	52.14	75.76	99.63	**84.57**	45.45	87.95	49.46	75.76	95.12	76.73	60.61	88.51	62.13
bn	**68.12**	39.38	80.18	53.03	33.15	59.21	65.15	38.65	74.94	46.97	34.16	61.18	65.15	**39.74**	**81.16**	53.03	36.16	64.57
mr	68.52	95.15	79.33	35.19	81.06	38.12	68.52	95.15	79.33	50.00	81.44	51.01	**70.37**	**95.58**	**79.57**	48.15	82.62	62.70

Table 3: Performance of model under federated settings conducted with 3 different client fractions. c is the fraction of clients whose updates are considered in every round. Evidently, for lower dataset sizes, $c = 10\%$ performs comparatively better.

	server-LSTM			**federated-LSTM**		
	Acc	AUC	F1	Acc	AUC	F1
te	51.28	73.91	54.60	66.67	74.32	54.38
hi	78.79	98.92	82.60	81.82	99.79	84.18
bn	60.61	40.92	83.46	68.12	39.38	80.18
mr	65.74	92.03	76.67	70.37	95.58	79.57

Table 4: Comparison between the centrally trained Bi-LSTM model and federated Bi-LSTM model.The federated model is selected based on best results from Table 3

Bengali (bn): Table 4 shows that the Federated Bengali model performs better in terms of accuracy against the centrally trained LSTM but worse in terms of AUC and F1-score. The reason is that the Bengali federated model is trained differently. For the synthetic-IID setting, every client cluster consists data from one specific dataset only and no dataset entries are mixed for every client. Since we use the combined dataset (Hasan et al., 2020), all of the datasets in it have different distributions in terms of categories. We believe that the poor AUC and F1 scores are due to this difference of distribution as the 'neutral' category is absent in some of these member datasets.

Marathi (mr): Looking at Table 4, it is evident that the Federated LSTM performs better on all metrics than the centrally trained LSTM. Since this trend continues across all the languages, we believe that Federated Learning helps learn the data representations better without data bottlenecks and also without sharing any data which might be sensitive.

6 Conclusion

We show that a LSTM model trained using federated learning can outperform an identical server trained LSTM model for 4 Indic languages - Marathi, Bengali, Telugu and Hindi. We also show that federated learning achieves comparable performance to other server trained Transformer architectures[4]. Surprisingly, we find that for smaller datasets, lower client fractions show better performance. To our knowledge, this represents one of the first applications of federated learning in low-resource settings for sentiment analysis. Federated learning offers security and privacy advantages for users by training across a population of highly distributed computing devices while simultaneously improving model performance.

For future work, it would be interesting to train heavier transformer models like IndicBERT, XLM-R, etc. using federated learning which could help to minimize the large gap in accuracy in non-iid settings. Conducting some interpretable evaluation on the intermediate models before updating during Federated training is another important future direction.

References

Gaurav Arora. 2020. iNLTK: Natural language toolkit for indic languages. In *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS)*, pages 66–71, Online. Association for Computational Linguistics.

Prafulla B Bafna and Jatinderkumar R Saini. 2021. Scaled document clustering and word cloud-based summarization on hindi corpus. In *Progress in Advanced Computing and Intelligent Engineering*, pages 398–408. Springer.

Zhexue Chen, Hong Huang, Bang Liu, Xuanhua Shi, and Hai Jin. 2021. Semantic and syntactic enhanced aspect sentiment triplet extraction. *arXiv preprint arXiv:2106.03315*.

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–

[4]The performance of IndicBERT to the federated LSTM is comparable and the lower results might be because of the difference in architecture complexity.

8451, Online. 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.

Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*.

Md Arid Hasan, Jannatul Tajrin, Shammur Absar Chowdhury, and Firoj Alam. 2020. Sentiment classification in bangla textual content: A comparative study. In *2020 23rd International Conference on Computer and Information Technology (ICCIT)*, pages 1–6. IEEE.

Mickel Hoang, Oskar Alija Bihorac, and Jacobo Rouces. 2019. Aspect-based sentiment analysis using BERT. In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pages 187–196, Turku, Finland. Linköping University Electronic Press.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Doaa Mohey El-Din Mohamed Hussein. 2018. A survey on sentiment analysis challenges. *Journal of King Saud University-Engineering Sciences*, 30(4):330–338.

Kushal Jain, Adwait Deshpande, Kumar Shridhar, Felix Laumann, and Ayushman Dash. 2020. Indic-transformers: An analysis of transformer language models for indian languages. *arXiv preprint arXiv:2011.02323*.

Aditya Joshi, AR Balamurali, Pushpak Bhattacharyya, et al. 2010. A fall-back strategy for sentiment analysis in hindi: a case study. *Proceedings of the 8th ICON*.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, NC Gokul, Avik Bhattacharyya, Mitesh M Khapra, and Pratyush Kumar. 2020a. inlpsuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for indian languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4948–4961.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. 2020b. IndicNLPSuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4948–4961, Online. Association for Computational Linguistics.

Md Rezaul Karim, Bharathi Raja Chakravarthi, John P McCrae, and Michael Cochez. 2020. Classification benchmarks for under-resourced bengali language based on multichannel convolutional-lstm network. In *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 390–399. IEEE.

Ankush Khandelwal, Sahil Swami, Syed Sarfaraz Akhtar, and Manish Shrivastava. 2018. Humor detection in english-hindi code-mixed social media content : Corpus and baseline system. *ArXiv*, abs/1806.05513.

Atharva Kulkarni, Meet Mandhane, Manali Likhitkar, Gayatri Kshirsagar, and Raviraj Joshi. 2021. L3CubeMahaSent: A Marathi tweet-based sentiment analysis dataset. In *Proceedings of the Eleventh Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 213–220, Online. Association for Computational Linguistics.

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).

Hao Li and Wei Lu. 2017. Learning latent sentiment scopes for entity-level sentiment analysis. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Hao Li and Wei Lu. 2019. Learning explicit and implicit structures for targeted sentiment analysis. *arXiv preprint arXiv:1909.07593*.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2018. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*.

Xin Li, Lidong Bing, Piji Li, and Wai Lam. 2019. A unified model for opinion target extraction and target sentiment prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6714–6721.

Bill Yuchen Lin, Chaoyang He, Zihang Zeng, Hulin Wang, Yufen Huang, Mahdi Soltanolkotabi, Xiang Ren, and Salman Avestimehr. 2021. Fednlp: A research platform for federated learning in natural language processing. *arXiv preprint arXiv:2104.08815*.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.

Sandeep Sricharan Mukku and Radhika Mamidi. 2017. ACTSA: Annotated corpus for Telugu sentiment analysis. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*, pages 54–58, Copenhagen, Denmark. Association for Computational Linguistics.

Joe Murphy, Michael W Link, Jennifer Hunter Childs, Casey Langer Tesfaye, Elizabeth Dean, Michael Stern, Josh Pasek, Jon Cohen, Mario Callegaro, and Paul Harwood. 2014. Social media in public opinion research: Executive summary of the aapor task force on emerging technologies in public opinion research. *Public Opinion Quarterly*, 78(4):788–794.

Braja Gopal Patra, Dipankar Das, Amitava Das, and Rajendra Prasath. 2015. Shared task on sentiment analysis in indian languages (sail) tweets-an overview. In *International Conference on Mining Intelligence and Knowledge Exploration*, pages 650–655. Springer.

Md Rahman, Emon Kumar Dey, et al. 2018. Datasets for aspect-based sentiment analysis in bangla and its baseline evaluation. *Data*, 3(2):15.

Yashwanth Reddy Regatte, Rama Rohit Reddy Gangula, and Radhika Mamidi. 2020. Dataset creation and evaluation of aspect based sentiment analysis in Telugu, a low resource language. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5017–5024, Marseille, France. European Language Resources Association.

General Data Protection Regulation. 2016. Regulation eu 2016/679 of the european parliament and of the council of 27 april 2016. Available at: `https://gdpr.eu/`.

Sebastian Ruder. 2020. Why You Should Do NLP Beyond English. `http://ruder.io/nlp-beyond-english`.

Gonzalo A Ruz, Pablo A Henríquez, and Aldo Mascareño. 2020. Sentiment analysis of twitter data during critical events through bayesian networks classifiers. *Future Generation Computer Systems*, 106:92–104.

S. M. Samiul Salehin, Rasel Miah, and Md Saiful Islam. 2020. A comparative sentiment analysis on bengali facebook posts. In *Proceedings of the International Conference on Computing Advancements*, ICCA 2020, New York, NY, USA. Association for Computing Machinery.

Apoorva Singh, Tanmay Sen, Sriparna Saha, and Mohammed Hasanuzzaman. 2021. Federated multi-task learning for complaint identification from social media data. In *Proceedings of the 32nd ACM Conference on Hypertext and Social Media*, HT '21, page 201–210, New York, NY, USA. Association for Computing Machinery.

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.

Mike Thelwall, Kevan Buckley, and Georgios Paltoglou. 2012. Sentiment strength detection for the social web. *Journal of the American Society for Information Science and Technology*, 63(1):163–173.

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.

Nafis Irtiza Tripto and Mohammed Eunus Ali. 2018. Detecting multilabel sentiment and emotions from bangla youtube comments. In *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pages 1–6. IEEE.

Jun Zhao, Kang Liu, and Liheng Xu. 2016. Sentiment analysis: mining opinions, sentiments, and emotions.

An Efficient BERT Based Approach to Detect Aggression and Misogyny

Sandip Dutta[†], **Utso Majumder**[†], **Sudip Kumar Naskar**[‡]
[†]Department of ETCE, [‡]Department of CSE
Jadavpur University
Kolkata, India
{sandip28dutta, utso1201, sudip.naskar}@gmail.com

Abstract

Social media is bustling with ever growing cases of trolling, aggression and hate. A huge amount of social media data is generated each day which is insurmountable for manual inspection. In this work, we propose an efficient and fast method to detect aggression and misogyny in social media texts. We use data from the Second Workshop on Trolling, Aggression and Cyber Bullying for our task. We employ a BERT based model to augment our data. Next we employ Tf-Idf and XGBoost for detecting aggression and misogyny. Our model achieves 0.73 and 0.85 Weighted F1 Scores on the 2 prediction tasks, which are comparable to the state of the art. However, the training time, model size and resource requirements of our model are drastically lower compared to the state of the art models, making our model useful for fast inference.

1 Introduction

With the rise of social media, there has also been a huge surge of online criticism and trolling. Mitigation of these kinds of online bullying has been an important problem and has been studied for a long time. However, the amount of information generated on social media is too large for humans to wade through. Machine Learning (ML) and Deep Learning (DL) models have achieved great success in the task of text categorization. However, even as larger models with higher accuracy are being built, the importance of the quality of data has still not reduced. Larger models will give sub-optimal results if the training data is not of good quality. But the results of large models are more difficult to interpret. Traditional ML models have the advantage of producing results that are relatively easier to interpret.

In this work, we combine a large deep learning model and traditional ML approaches for the purpose of text classification. Our experimental data is comprised of social media texts from YouTube comments and the task is to predict the presence of aggression and misogyny from the data. First the data is analyzed to reveal the class distribution issue with the data. Next we develop a Bidirectional Encoder Representations from Transformer (BERT) (Devlin et al., 2019) based text data augmentation pipeline to fix the class distribution. This augmented data helps to reduce class imbalance. For the classification part, we develop a Tf-Idf and XGBoost based classification model to classify the text.

Our model[1] achieves a score very close to the state of the art. However, since the augmentation task is essentially a one-time process, our model is simpler and faster. The main classification pipeline does not require high-end computational resources during inference, which makes our model even more efficient.

2 Related Works

So far, significant contributions have been made in the domain of aggression detection in text (Razavi et al., 2010; Kumar et al., 2018, 2020b). Other aspects of of the task have been investigated in areas such as trolling (Cambria et al., 2010; de la Vega and Ng, 2018), misogyny (Anzovino et al., 2018), cyberbullying (Dadvar et al., 2013; Xu et al., 2012), racism (Greevy, 2004), offensive language(Nobata et al., 2016) and hate speech(Djuric et al., 2015; Davidson et al., 2017). All these works are diverse in terms of the target subject they investigate. The works have mainly been conducted on English datasets, however there are some other languages on which works have been reported, for example, in Hindi (Mandla et al., 2021),Spanish (Garibo i Orts, 2019),Chinese (Su et al., 2017), etc.

[1]Source Code : https://github.com/Dutta-SD/AggDetect

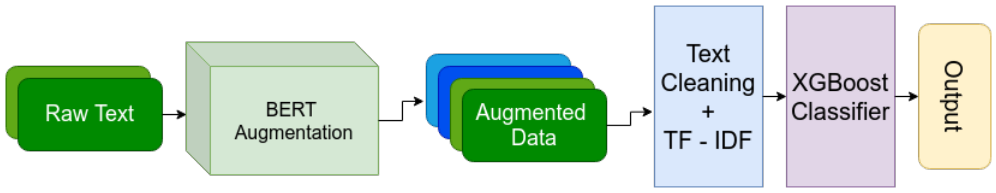


Figure 1: Model Pipeline

3 Model Description

3.1 BERT based Text Data Augmentation

The dataset had a skewed distribution, with the majority of the data belonging to one category. This imbalance can cause models to always predict the majority class. To counter this effect, we employed a BERT based masked word prediction model for data augmentation. The process of augmentation is presented in Algorithm (1).

Algorithm 1 BERT based Data Augmentation - Masked Word Prediction

```
1: Select one text belonging to the minority class.
2: Filter out the stop words from the text as they do not contribute to the overall sentiment of the text.
3: Randomly select one token and replace it with a special [MASK] token. The model fills this token with the word that it deems to be most likely in that context.
4: We take the top k predictions for augmenting, where 'k' is a hyper parameter. We choose k as per the dataset to ensure equal class distribution.
5: Repeat the above steps for other texts in the minority class to make the distribution even for all the classes.
```

The dataset is augmented once and stored. This essentially being an offline one time process, makes our overall model faster. This process need not be repeated during inference, which makes our net model size smaller.

3.2 Text Data Cleaning

To clean the texts, we use the following steps:

- **Remove Punctuation** – Punctuation marks are replaced with empty string.
- **Remove Stop Words** – We use the default stop words list of `nltk` package and replace the stop words with the null character. We exclude words like "no", "not", "ain't" since they change the semantic nature of a sentence; therefore these words are retained.
- **Stemming** – `Porter Stemmer` from the `nltk` package is used to reduce the vocabulary size and noise.

3.3 Tf–Idf Vectorization

Tf–Idf Vectorization is a scheme that converts a given text into a vector of numerical values. Classification task is made easier by using Tf–Idf, since taking the log of the inverse count of t term reduces the value of unimportant words occurring more frequently in the document. Tf–Idf returns a sparse vector for each text. This helps reduce memory consumption and training time.

3.4 XGBoost Text Classifier

XGBoost (Chen and Guestrin, 2016) or Extreme Gradient Boosting is a very popular algorithm for classification. The model is preferred because of the following reasons.

- **Sparsity Aware Computation** – XGBoost supports computations on sparse data, which helps avoid unnecessary overheads and results in faster training.
- **Approximate Splitting Algorithm** – BERT based data augmentation (cf. Section 3.1) of our pipeline increases the amount of data in our dataset by data augmentation. XGBoost provides approximate methods for splitting the data which saves memory.
- **Regularized Learning Objective** – XGBoost provides regularized objective function which controls model complexity and prevents over-fitting.

4 Experiments and Results

4.1 Dataset

The English dataset (Bhattacharya et al., 2020) we used for our experiments consists of texts scraped from YouTube comments section (Kumar et al.,

Table 1: Experiments on Various Classifiers and Weighted F1 Scores (Other Hyper-parameters are Default Values)

Model	**Sub Task A Score**	**Sub Task B Score**
Multi Layer Perceptron Classifier (50 iterations)	0.533	0.720
Random Forest Classifier	0.561	0.740
LinearSVC	0.620	0.773
XGBoost Classifier ($\gamma = 0.2$)	0.729	0.850
Final Model – XGBoost Classifier ($\gamma = 0.1$)	**0.735**	**0.852**

Table 2: Results obtained (Weighted F1 Score)

Team Name	**Sub Task A Score**	**Sub Task B Score**
Julian (Risch et al., 2020)	0.802	0.851
abaruah (Baruah et al., 2020)	0.728	0.870
sdhanshu (Safi Samghabadi et al., 2020)	0.759	0.857
Our best model	**0.735**	**0.852**

2020a). The task consists of 2 sub tasks - 'Sub Task A' and 'Sub Task B'.

Sub Task A is to detect the level of aggression in the text and for this sub task the dataset contains 3,375 Non Aggressive (NAG), 453 Covertly Aggressive (CAG, Disguised aggressive content e.g, sarcasm), and 435 Overly Aggressive (OAG, directly aggressive) samples.

Sub Task B addresses misogyny identification and for this the dataset comes with 3,954 Non Misogynistic (NGEN) and 309 Misogynistic (GEN) samples.

An additional validation set was provided with 836 NAG, 117 CAG and 113 OAG samples for Sub Task A and 993 NGEN and 73 GEN samples for Sub Task B. The testset comprised of 690 NAG, 286 OAG and 224 CAG samples for Sub Task A and 1,025 NGEN and 175 GEN samples for Sub Task B.

For both the sub tasks, the ratio of samples indicates a huge data imbalance. This was fixed by using BERT based augmentation and then prediction was done using the model as described in Section 3.2 – 3.4.

4.2 Experimental Setup

Figure 1 shows the entire classification pipeline. Hyper-parameter `k` (cf. Algorithm 1) was set to 2 for our model to augment each data point 2 times. This entire process was repeated 2 times to even out the distribution.

A number of experiments were performed on Random Forest, SVM and Multi Layer Perceptron using scikit-learn (Pedregosa et al., 2011). The BERT masked word prediction augmentation model was moved to the GPU but not finetuned. All hyper-parameters were set as per Devlin et al. (2018). It took about 10 minutes to finish the entire data augmentation process on a Nvidia Tesla K80 GPU.

Next, the Tf-Idf + XgBoost pipeline was trained on an Intel Xeon CPU. The `gamma` hyperparameter of XGBoost model was set to 0.1 for regularization. All other hyper parameters were kept to their default values. The entire process of cleaning, training, validation and predicting on test data took about 2 minutes to complete. We evaluated our models using weighted F1 score.

4.3 Results

The results of our experiments are reported in Table 1. Among our models, XgBoost with $\gamma = 0.1$ yielded the best results for both the subtasks.

Table 2 presents a comparison of our best model with the best models on this task reported in the literature. Risch et al. (2020) obtained the highest scores on Sub Task A and they used bagging of multiple BERT models for prediction. Their entire pipeline required 7 hours to train on a Nvidia 1080 Ti GPU. Baruah et al. (2020) reported the best results on the Sub Task B and they used Transformer based models which demand heavy computational resources. Another top performing model (Safi Samghabadi et al., 2020) took about 6 hours to train on an Nvidia Tesla P40 GPU.

Our model achieves nearly comparable results to the state of the art for this task but requires only a small fraction of the computational resources and time that the top models need. To train the model, the total time required is about 30 minutes, which

Table 3: Predictions by Our Model (Some Texts are Truncated)

Text	**Aggression**		**Misogyny**	
	Actual	**Predicted**	**Actual**	**Predicted**
watch sandeep's interview in film companion fat guy with salt and pepper hair (rajiv masand)	NAG	CAG	GEN	NGEN
maria malhotra yes, but i am more of dying on the interviewers face. it looked like a square. I was rolling on the ground	NAG	OAG	NGEN	NGEN
sesh r kotha gulo just heart touching....	NAG	NAG	NGEN	NGEN
where the hell is that sexuall predater cunt. she must be exposed by judiciary, media, parliament	OAG	OAG	NGEN	GEN

Table 4: Model performance (Weighted F1 Score) with different values of hyperparameter `k`

Value of k	**Sub Task A Score**	**Sub Task B Score**
0 (No Augmentation)	0.232	0.354
2 (Our Experiment)	0.735	0.852
4	0.551	0.672

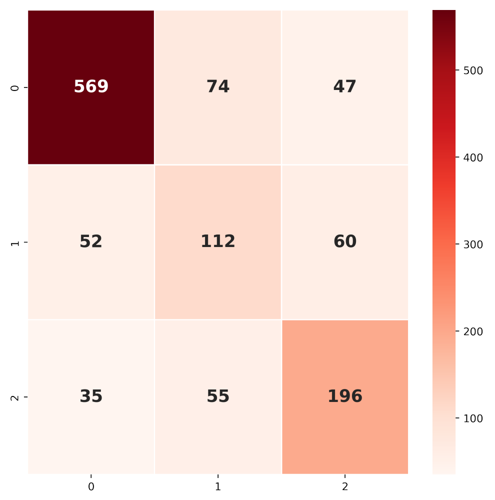


Figure 2: Confusion Matrix Sub Task A (0: NAG, 1: CAG, 2: OAG, Y-Axis: True Label; X-Axis: Predicted Label)

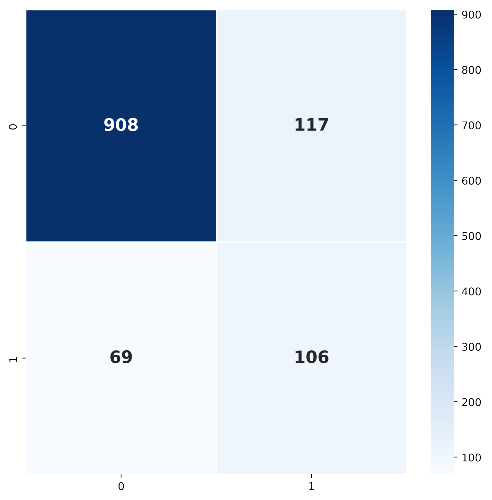


Figure 3: Confusion Matrix Sub Task B (0: NGEN, 1: GEN, Y-Axis: True Label; X-Axis: Predicted Label)

is drastically less compared to the state of the art. The storage requirement for our model is also lower. Our models occupy about few MBs (`.pkl` files), whereas other models take hundreds of MBs.

4.4 Analysis

Table 4 shows the performance of the classifier based on different values of the hyperparameter `k`. It is evident from the table that with no augmentation (`k` = 0), the model performs poorly. With `k` = 2, which makes the class distribution almost even, the model gives the best performance. For higher values of `k`, the class distribution becomes skewed and we get deteriorated performance. Thus best performance is obtained when there is almost equal distribution of all the classes; the model can then learn better representations of the data.

Figure 2 and Figure 3 show the confusion matrix for Sub Task A (Aggression Detection) and Sub Task B (Misogyny Detection), respectively.

We see that our model detects non aggressive and overtly aggressive categories well. However, it shows some problem in detecting aggression in covert forms such as in sarcasm. For misogyny detection, our model shows some errors.

Table 3 shows some predictions from our model where the model predictions match or differ from the actual labels. As has also been reported in Safi Samghabadi et al. (2020), there are some labelling errors in the dataset itself. Our model predictions seem to be more appropriate than ground truth values in those cases.

5 Conclusion

In this work, a fast and efficient pipeline to detect aggression and misogyny from social media texts was developed. The text data is cleaned and analysed and data augmentation was performed using a BERT based model. Various models were experimented upon and Tf-Idf + XgBoost model performs the best among them. Weighted F1 Score of 0.735 and 0.852 was produced by our model, however with drastically reduced computational requirements.

In future, we aim to perform the training on a larger dataset to obtain better results. This would enable us to develop a fast and efficient system to tackle the problem of hatred on social media sites. There is scope of improvement in specific areas like improving the pipeline to detect the subcategory of covertly aggressive better, and to detect misogynous content slightly better. We can extend the reach of the present work to processing multilingual datasets as well.

References

Maria Anzovino, Elisabetta Fersini, and Paolo Rosso. 2018. Automatic identification and classification of misogynistic language on twitter. In *International Conference on Applications of Natural Language to Information Systems*, pages 57–64. Springer.

Arup Baruah, Kaushik Das, Ferdous Barbhuiya, and Kuntal Dey. 2020. Aggression identification in English, Hindi and Bangla text using BERT, RoBERTa and SVM. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 76–82, Marseille, France. European Language Resources Association (ELRA).

Shiladitya Bhattacharya, Siddharth Singh, Ritesh Kumar, Akanksha Bansal, Akash Bhagat, Yogesh Dawer, Bornini Lahiri, and Atul Kr. Ojha. 2020. Developing a multilingual annotated corpus of misogyny and aggression. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 158–168, Marseille, France. European Language Resources Association (ELRA).

Erik Cambria, Praphul Chandra, Avinash Sharma, and Amir Hussain. 2010. Do not feel the trolls. *ISWC, Shanghai*.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Maral Dadvar, Dolf Trieschnigg, Roeland Ordelman, and Franciska de Jong. 2013. Improving cyberbullying detection with user context. In *European Conference on Information Retrieval*, pages 693–696. Springer.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language.

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. pages 29–30.

Edel Greevy. 2004. *Automatic text categorisation of racist webpages*. Ph.D. thesis, Dublin City University.

Ritesh Kumar, Guggilla Bhanodai, Rajendra Pamula, and Maheshwar Reddy Chennuru. 2018. Trac-1 shared task on aggression identification: Iit (ism)@ coling'18. In *Proceedings of the first workshop on trolling, aggression and cyberbullying (TRAC-2018)*, pages 58–65.

Ritesh Kumar, Atul Kr. Ojha, Bornini Lahiri, Marcos Zampieri, Shervin Malmasi, Vanessa Murdock, and Daniel Kadar, editors. 2020a. *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*. European Language Resources Association (ELRA), Marseille, France.

Ritesh Kumar, Atul Kr Ojha, Shervin Malmasi, and Marcos Zampieri. 2020b. Evaluating aggression identification in social media. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 1–5.

Thomas Mandla, Sandip Modha, Gautam Kishore Shahi, Amit Kumar Jaiswal, Durgesh Nandini, Daksh Patel, Prasenjit Majumder, and Johannes Schäfer. 2021. Overview of the hasoc track at fire 2020: Hate speech and offensive content identification in indo-european languages.

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.

Òscar Garibo i Orts. 2019. Multilingual detection of hate speech against immigrants and women in Twitter at SemEval-2019 task 5: Frequency analysis interpolation for hate in speech detection. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 460–463, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

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.

Amir H Razavi, Diana Inkpen, Sasha Uritsky, and Stan Matwin. 2010. Offensive language detection using multi-level classification. In *Canadian Conference on Artificial Intelligence*, pages 16–27. Springer.

Julian Risch, Robin Ruff, and Ralf Krestel. 2020. Offensive language detection explained. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying (TRAC@LREC)*, pages 137–143. European Language Resources Association (ELRA).

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 (TRAC-2020)*, pages 126–131, Marseille, France. European Language Resources Association (ELRA).

Hui-Po Su, Zhen-Jie Huang, Hao-Tsung Chang, and Chuan-Jie Lin. 2017. Rephrasing profanity in Chinese text. In *Proceedings of the First Workshop on Abusive Language Online*, pages 18–24, Vancouver, BC, Canada. Association for Computational Linguistics.

Luis Gerardo Mojica de la Vega and Vincent Ng. 2018. Modeling trolling in social media conversations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012. Learning from bullying traces in social media. In *Proceedings of the 2012 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 656–666.

How vulnerable are you? A Novel Computational Psycholinguistic Analysis for Phishing Influence Detection

Anik Chatterjee
St Thomas College of
Engineering and Technology
Kolkata, India
anikchatterjee63@gmail.com

Sagnik Basu
St Thomas College of
Engineering and Technology
Kolkata, India
sagnikbasu19@gmail.com

Abstract

This document contains our work and progress regarding phishing detection by searching for proper influential sentences. Currently, the world is becoming smart, as a result most of the transactions and posting offers happen online. So, human beings have become the most vulnerable to security breach or hacking through phishing attacks, or being persuaded through influential texts in social media sites. We have analyzed influential and non-influential sentences and populated our dataset with those. We have proposed a computational model for implementing Cialdini and we got state of the art accuracy with our model. Our approach is language independent and domain independent and it is applicable to any problem where persuation detection is important. Our dataset and proposed computational psycholinguistic approach will motivate researchers to work more in the area of persuasion detection.

1 Introduction

In some ways we humans are incredibly durable, and in others we are incredibly fragile. Most of the humans can be easily swayed by beautiful rewards. We are the most vulnerable to phishing attacks and such rackets. Some people take these opportunities to persuade people to click some bad links through some deceitful messages.

Susceptibility (Pierre O. Jacquet, 2018), persuasion (MakuochiNkwo and Orji, 2018; Kiemute Oyibo and JulitaVassileva, 2018; Ifeoma Adaji and JulitaVassileva, 2020; Christopher Hidey, 2018) and gullibility (Mercier, 2017), these three words are very much related to human behaviour. Humans are susceptible to treachery, they can be persuaded easily, and they are gullible too to believe anything in what others say.

Depending on a person's behaviour we can influence him/her using different types of methods. Robert Cialdini did research on all those methods and ways and published his famous principles. We created a classifier to detect types of influence by fine-tuning a pre-trained BERT model.

In this paper our major contributions are:

i. Our research on phishing detection was using computational psycholinguistic approach. Here we have modeled the Cialdini principles on influence analysis.

ii. We have prepared our dataset using influential texts, such as advertisements, Twitter posts etc.

iii. We experimented with pre-trained BERT models by adding extra layers and changing the learning rate.

The rest of the paper is organised as follows. Section 2 discusses about the related works and how those are different. Section 3 discusses about Cialdini principles and how we applied them in our work. Section 4 describes how we made our own dataset and related literature survey. Section 5 provides an insight into the pre-trained model and why we used it. Section 6 discusses about our experiment with different methods and tools to reach higher accuracy. Section 7 provides a detail about our result and section 8 concludes the paper.

2 Related Work

The closest research to our work is on sentiment analysis and persuasion detection based on Cialdini principles. Kiemute Oyibo and JulitaVassileva (2018) focused on how culture and gender influence the effectiveness of Cialdini's principles of persuasion. They investigated on how the culture, gender and age differences affect individual's susceptibility to Cialdini's persuasive strategies. But their work centred on the people of Nigeria. Sridhar Ramaswamy (2018) used unstructured data available for survey voice recording of customer interactions and chat transcripts and explored different technologies of Deep Learning (Goodfellow et al., 2016) and Natural Language Processing (NLP) that would help better analyze the contextual information to capture customer feedback. Christopher Hidey (2018) worked with detection of persuasion in online discussion or in some argument. Merton Lansley (2019) developed a method that detects social engineering attacks that are based on NLP and Artificial Neural Networks. Ifeoma Adaji and JulitaVassileva (2020) proposed the use of shoppers' online shopping motivation in tailoring six commonly used influence strategies: scarcity, authority, consensus, liking, reciprocity, and commitment and they identified how these influence strategies can be tailored or personalized to e-commerce shoppers based on the online customers' motivation when shopping. Shilpa P C (2021) used deep learning techniques to classify the sentiments of an expression into positive or negative emotions which were further classified into more atomic emotions. Vansh Gupta (2021) worked with persuasion detection but their work was based on biased or misleading information or propaganda classification.

Some researchers have also worked on sentiment analysis and persuasion detection based on Twitter data. OlhaKaminska and Hoste (2021) developed an approach for the SemEval-2018 emotion detection task, based on the Fuzzy Rough Nearest Neighbor (FRNN) classifier enhanced with Ordered Weighted Average (OWA) operators. Neha Jadav and Khamparia (2018) focused more on improvement of accuracy of sentiment system 1 to 5 star, 1 being the most negative.

So many works on persuasion detection have been done over the years. Everyone has focused on some particular domain, either customer feedbacks, online shopping sites, or some particular culture or area. Our goal was to identify phishing attacks through influential texts. Therefore we used persuasion detection and Cialdini principles and we tried to be more general in determining effects of these influences. So, we collected example sentences from as many diverse places as possible.

3 Our Technique

While searching for the ways by which people can be persuaded or phishing attacks can be detected we came across Cialdini's principles.[1] So we used mainly those principles in our classifier. Before going further let's first discuss about Cialdini principles.

3.1 Cialdini's Principles

Robert Cialdini published his book *"Influence: The Psychology of Persuasion"*[2] in 1984. In this book he explored some factors that affect the decisions that people make. Cialdini identified six core principles that affect this decision making process.

Reciprocity — People always tend to help those from whom they have got help before as a form of gratitude. So business companies make the advertisements in a way to provide their customers extra benefits, discounts, offers in order to prompt them to buy products from that company.

Scarcity — It is a fact that the less something there is, the more people will tend to want it. Many companies use this human behaviour to put some products in limited edition sale.

Authority — Individuals who are authoritative, credible, expert in their fields are more influential and persuasive than those who are not. People prefer to go for those company products that are promoted by authoritative figures.

Commitment and Consistency — People like them who are committed to their words, and they also like to be consistent with their identity. So many advertisements tend to make people believe something and make them do according to that. People also like to use the products which are committed to their usefulness.

Liking — People tend to follow those whom they love, that could be an authoritative person, a player, a singer. So some companies give such famous persons money to use their products and show to people.

Consensus — People love opinion of the majority. When they find something is used by most of the people they tend to use it too.

[1]https://worldofwork.io/2019/07/cialdinis-6-principles-of-persuasion/
[2]https://www.goodreads.com/book/show/28815.Influence

3.2 How we applied Cialdini theory

We used this Cialdini Persuasion theory in order to find the influence working behind a text, be it an advertisement or a text message. We made each category a class and thus we applied Cialdini's Principles as a multiclass Machine Learning (Mitchell, 1997) classification problem.

We chose Cialdini's principles to divide texts in multiple classes. We marked those classes as 1) Reciprocity, 2) Scarcity, 3) Authority, 4) Commitment, 5) Liking, 6) Consensus and we created a 7th class as not showing any influence, for normal sentences.

We collected email texts from different senders and went through different ads and categorized them according to the levels of persuasion.

Phishing attacks target vulnerabilities that exist in systems due to the human factor (Khonji et al., 2013). Many cyber attacks are spread via mechanisms that exploit weaknesses found in end-users, which makes users the weakest element in the security chain. The phishing problem is broad and no single silver-bullet solution exists to mitigate all the vulnerabilities effectively, thus multiple techniques are often implemented to mitigate specific attacks. One of these techniques is persuasion analysis.

4 Making Dataset

We studied Cialdini's theory to understand how a person can be influenced in order to detect phishing attacks. We did not find any dataset regarding our research, so we had to make our own dataset and categorize the texts by ourselves. We used some pre-defined definitions in order to sort those sentences out. For identifying the six influence classes, we followed the following conventions —

- Give away or giving something without any charge on some occasion, such messages show reciprocity type of influence.
- Messages alerting customers regarding limited offer show scarcity type of influence.
- Messages or ads regarding sponsorship or doing partnerships show authority type of influence.
- Messages committing about well being or best performance of a product show commitment type of influence.
- Messages telling what one's favourite person uses or does, provoking him/her to use that product show liking type of influence.
- Messages showing survey results show consensus type of influence.

We collected as many influential texts as possible from advertisements from different companies and grouped them in those classes. We gathered about 100 texts from each class and made a dataset with 735 texts. We created a model using this dataset.[3]

Still the variety in our dataset was very small. So we collected more examples using twitter scraping of companies like – Amazon, Tesla, Microsoft, Flipkart, Tinder etc. From Amazon, Flipkart, we got more examples on scarcity and reciprocity. From Tesla, we got more examples on commitment types. Tinder gave us examples on liking. Big companies like these are also perfect for finding examples on authority and consensus. These sentences helped us understand how social sites can persuade people through influential advertisements and messages. After collecting these sentences on various domains we used our temporary model to make predictions. Then we rechecked, verified and corrected the predictions that were made above 95% and discarded others. Finally we appended those texts with our original dataset (please see table 1) and made its size 2379.

5 Used model for classification

5.1 Transformers vs RNNs

Transformers (Ashish Vaswani and IlliaPolosukhin, 2017) are semi-supervised machine learning models that were primarily used with text data and have also replaced Recurrent Neural Networks (RNNs) (Danilo P. Mandic, 2001) in Natural Language Processing (NLP) (Jurafsky and Martin, 2009) tasks. We chose Transformers for our problem because it beats RNN in time complexity.

Transformers use self-attention mechanism that allows the decoder to look back at the entire sentence and selectively extract the information it needs during decoding. This mechanism helps to know the context better. With RNN, one has to go word by word to access the cell of the last word. This becomes a major problem for GPUs,

[3]https://github.com/starboi2000/Phishing-detection-and-influence-analysis/blob/main/previous_data.xlsx

Sentence	Influence Class Number	Influence Class Name
@lifeisahandful You're most welcome! We're happy we can help brighten your day!	1	Reciprocity
@RachelLivesLife Thanks for the support! We are hiring, have a look.	1	Reciprocity
@JaggiPagal Apologies for the unpleasant experience with your order. Could you confirm if we've missed the estimated delivery date?	2	Scarcity
@FunSizeDel Time to bust out those happy moves, and dance All Night Long! Enjoy your order! #deliveringsmiles	2	Scarcity
@nhuebecker Alexa play The Fame by Lady Gaga	3	Authority
You guessed it - it's their next look! With the latest trends and top brands at Flipkart Fashion - India Ka Fashion Capital, there's no other way than to Wear the next!	3	Authority
You can now contribute to on-ground COVID relief services with #Check-OutGiving . Donate Rs. 10 to @GiveIndia when you check out with just a click of a button. #FlipkartCares #FlipkartForIndia.	4	Commitment
If you're not 100% in love after the first 30 nights, we'll pick it up, do all the packing, and give you a full refund. We do our best to donate returned mattresses and give them a new home.	4	Commitment
@StephTheGroupie You've got to be squidding! We're shrimply lobsessed with these slides. A pair of these would make anyone jelly(fish)!	5	Liking
@cutenfeisty– There's nothing quite like a good book journey! Tell us, what's your favorite genre of books to read on your Amazon Kindle device?	5	Liking
for the last 10 years we are the number 1 in this industry by our customer review	6	Consensus
57% of consumers will buy this or use a this service because it has at least a 4-star rating.	6	Consensus
Having Zelda's approval on the box just brightened up our Caturday! What does she enjoy doing when she's not lounging?	7	Normal
@pww3777 Someone sure looks comfurtable! What's this little cutie's name?	7	Normal

Table 1: Example Dataset

this sequentiality is an obstacle to the parallelization of the process. Whereas, transformer proposes to encode each position and to apply the mechanism of attention in order to connect two distant words, which can then be parallelized, accelerating learning.(Louis-Philippe Morency, 2018)

5.2 BERT vs Others

More than one architectures are being used in NLP tasks, such as ELMo, GPT, BERT.(Jacob Devlin, 2019)

ELMo follows a more traditional design and uses LSTM to compute vocabulary when it comes to art. BERT uses transformers in two approaches for language improvement. ELMo uses two layers each to include forward and backward passages to calculate the middle word vectors. This helps ELMo to achieve high efficiency compared to other traditional language models.(Ezen-Can, 2020)

Though transformer uses a decoder and an encoder, BERT only uses an encoder and it does bi-directional context search, whereas GPT does it in one direction. BERT can also be used for multi-masked sentences, and next sentence prediction, and bi-directional search really makes it more reliable. Moreover, BERT model reads at most 512 words in one iteration. GPT-3 has almost 175B parameters and T5 has 11B parameters, whereas

BERT has only 110M parameters in its base model and 340M parameters in its large model. Based on our subject we did not require so many parameters as GPT and T5 provide, that is why we preferred BERT over other models.

We picked BERT-base-uncased over cased as influence detection should not be sensitive to cases.

6 Experiments with different methods

We trained a wide range of different models for the task. As discussed in sections 3 and 4 we made our own dataset and used them to train the models. As we discussed in section 5, we used BERT model from transformers rather than RNN.

After making the train and validation datasets we applied different pre-trained models, fine-tuned them and used the best one to make our final model.

We fine tuned BERT-base-uncased with our Neural Network which is taking input from the fine tuned BERT layer and processing it in this extra 9 layers out of which first 7 layers are simple dense layers with the unit numbers of 1024, 512, 256, 128, 128, 64, 32, a dropout layer with 0.3 dropout rate and lastly an output dense layer with unit number of 7.

We fine tuned another BERT-base-uncased model with previous configurations but this time without the dropout layer. We wanted to check how it behaves after removing the dropout layer.

Then we also fine tuned a BERT-base-cased model with a total of 9 layers among which 7 layers are simple dense layers with unit numbers 1024, 512, 256, 128, 128, 64, 32, an output dense layer with unit number of 7 and one dropout layer with rate 0.3. We checked by removing the dropout layer also for this model.

Then we also fine tuned a Distilled version of BERT model or DistilBERT.

Our Deep Neural Network model used a vocabulary size of 10,000, a batch size of 32 and was trained over 10 epochs. The system consisted of two input layers, one main BERT layer, six dense layers with varying sizes decreasing from 1024 to 32, one dropout layer with regularization of 0.3 and one output layer.

We used Rectified Linear Unit (ReLU)[4] activation function for dense layers and Softmax function for the output layer. The main advantage of using ReLU over other activation functions is that it does not activate all the neurons at the same time. This means that the neurons will only be deactivated if the output of the linear transformation is less than 0.

We experimented with all these models using different learning rates, because not only the method, but how fast the model is learning, is also very crucial in gaining more accuracy. We started with 0.01 and decreased the rate each time until we got the maximum accuracy. Then we collected the learning rate and accuracy for each model. Among all these models, we got the best result in BERT-base-uncased without dropout layer model with learning rate of 0.0001 or 1e-4.[5]

7 Result and discussion

There are many datasets on whether a sentence is ham or spam[6][7] and datasets on detecting phishing sites[8](Justinas Rastenis, 2021; Patrick Lawson, 2020; Kiemute Oyibo and JulitaVassileva, 2018). But there is no dataset on influence analysis using Cialdini principles, so we had to make our dataset. Then we experimented with different Deep Neural Network models and found BERT-base-uncased to overwhelm others and be suited for our work.

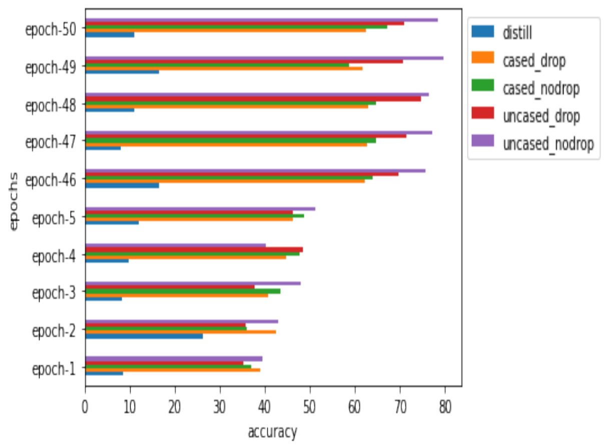


Figure 1: Accuracy Plot

Figure 1 shows epoch count versus accuracy for different models. We used the transformer models

[4]https://bit.ly/32YjPlx

[5]https://github.com/starboi2000/Phishing-detection-and-influence-analysis/blob/main/All_Codes_in_ipynb_to_understand_better/Cialdini_6_Principles.ipynb

[6]https://github.com/laxmimerit/All-CSV-ML-Data-Files-Download

[7]https://bit.ly/3lpxlFi

[8]https://bit.ly/32InPX6

DistillBERT, BERT-base-cased and uncased both of them with or without dropout. Among them we got the best result in BERT-base-uncased without dropout layer, we can clearly see in the graph above that the violet line is exceeding other lines in almost every epoch.

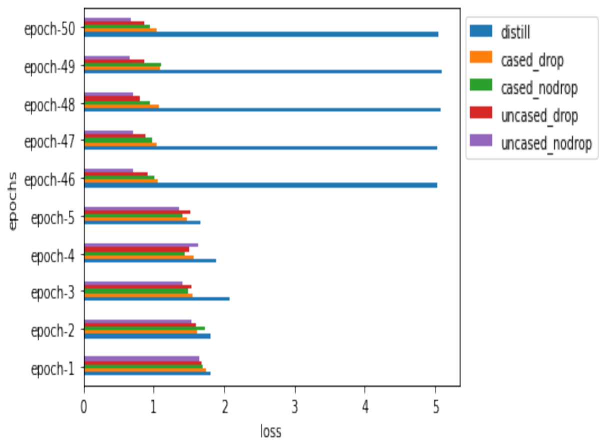


Figure 2: Loss Plot

Figure 2 shows epoch count versus loss for different models. As we got the best accuracy in accuracy graph for BERT-base-uncased without dropout, similarly we got the least loss in case of this model.

We used Bert-base-uncased model of Transformer to make our model with a learning rate of 0.0001 as it reached the highest accuracy. We achieved 97.46% accuracy with our small dataset. After appending it with more example texts, we got 93.76% from our model using the new dataset.[9]

We have also kept a sub-part[10] of the dataset, as our test data, and used that to evaluate our model. It turned out to be 87.5%.

Our model does not show which type of influence is being used, but rather it shows each type of influence along with their probabilities of being present in the given text.

Figure 3 shows the prediction of our proposed model by displaying the probabilities of each influence category in a given sentence.

To check this model is predicting well in the dataset, we used confusion matrix (See figure 4).

True Positive, False Negative, False Positive, True Negative for each class are shown in Table 3.

[9]https://github.com/starboi2000/Phishing-detection-and-influence-analysis/blob/main/main_data.xlsx

[10]https://bit.ly/3IcxPbs

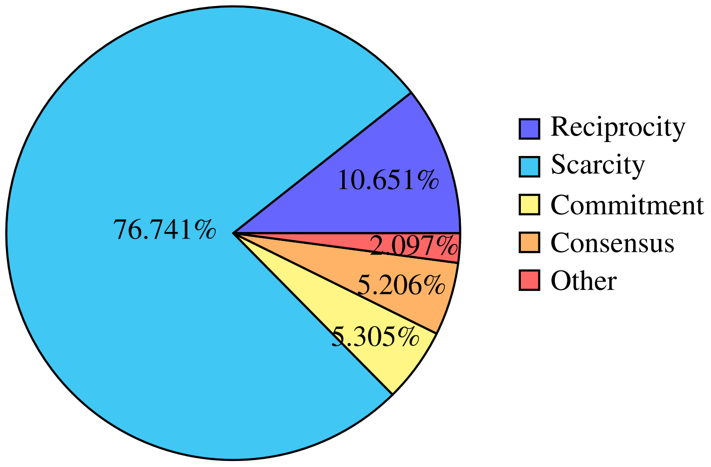


Figure 3: An example of the output

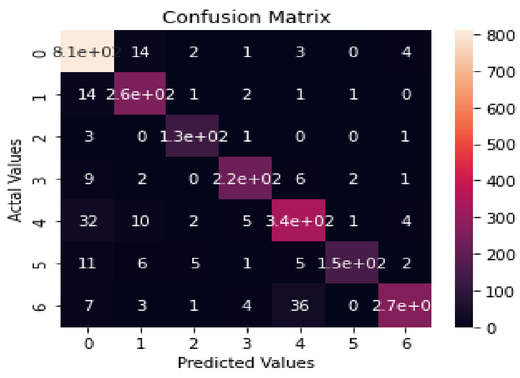


Figure 4: Confusion Matrix

Total number of true positive in 7 classes is 2176 out of 2379. We can see that we have the most information on Reciprocity type of influence and the least on Authority and Social Proof. As we have scraped business data, Reciprocity is the most dominant one there, and then comes Commitment. To make the understanding better we use F1-score which depends on both recall and precision.

In Table 2, we can check the F1-scores for different models on each class. We checked running all the models for 50 epochs. We can see that how we are getting the highest accuracy of 78% with the BERT-base-uncased model without a dropout layer.

8 Conclusion

Influence detection is a multi-level classification problem. Many works have been done on this topic and on persuasion detection, but they have worked on a particular domain, area, or caste. We focused more on finding the example texts not only from

	Reciprocity	Scarcity	Authority	Commitment	Liking	Consensus	Normal	Accuracy
Cased with dropout	0.81	0.66	0.42	0.44	0.52	0.25	0.6	0.63
Cased without dropout	0.83	0.65	0.39	0.6	0.6	0.3	0.67	0.67
Uncased with dropout	0.81	0.78	0.61	0.65	0.6	0.4	0.68	0.71
Uncased without dropout	0.87	0.8	0.7	0.77	0.7	0.65	0.76	0.78
Distil BERT	0.69	0.54	0.48	0.6	0.65	0.35	0.6	0.61

Table 2: F1-Score

	TP	FN	FP	TN
Class-0	810	24	76	1469
Class-1	258	19	35	2067
Class-2	127	5	11	2236
Class-3	224	20	14	2121
Class-4	338	54	51	1936
Class-5	153	30	4	2192
Class-6	266	51	12	2050

Table 3: TP, FN, FP and TN for each class

advertisements, but from normal messaging texts also.

We focused on detecting phishing and other social media attacks, because these are something to which people are the most vulnerable nowadays. We made our own dataset and we have not predicted only the type of influence; we have given all the probabilities for each type of influence. Even for a human, we cannot predict exactly what type of influence is working in a sentence. Sometimes, a sentence can be formed using more than one influence categories. Advertisements include multiple influences in one sentence, so that becomes very difficult for anyone to determine the most effective type of the sentence. So, we have shown the probabilities as a pie chart (please see figure 3) and also mentioned the type of influence that is most probably working.

Where from Here

i. At first we did not find any suitable dataset for our work, so we collected data from different advertisements and sentences from Twitter and we made our model. The final dataset that we have made is still not sufficient, so our model may be overfitted. We plan to increase the data more in the future using Bootstrap mechanism. Then by human intervention we will verify and rectify the predictions made by our model and train it with new data.

ii. We have fine tuned cased and uncased versions of pre-trained BERT model and we have also used DistilBERT. As our future work, we want to do more research using recent pre-trained models for better results.

iii. We have applied phishing detection using influence analysis in English language only. However, our proposed approach is domain and language independent. Therefore we are planning to apply our model for other Indian Languages starting from Bangla. We are also planning to see how it will work for fake news detection problem where influence detection is also important.

iv. Limited priorwork has been done on phishing detection using persuasion techniques. We did not have the opportunity to compare with those datasets. As a future work, we are planning to compare our dataset with different datasets and different methodologies.

References

Robert B. Cialdini 2004. The science of persuasion. In *Scientific American Mind 284, (2004), 76-84.*

Soumya Sahoo Ambik Mitra, Lambodar Jena. 2021. Emoji analysis using deep learning, advances in intelligent computing and communication (pp.689-698).

Niki Parmar Jakob Uszkoreit Llion Jones Aidan N. Gomez Lukasz Kaiser Ashish Vaswani, Noam Shazeer and IlliaPolosukhin. 2017. Attention is all you need. In *Conference on Neural Information Processing Systems*.

Kathleen McKeown Christopher Hidey. 2018. Persuasive influence detection: The role of argument sequencing. In *Proceedings of the AAAI Conference on Artificial Intelligence, 32(1)*.

Jonathon A. Chambers Danilo P. Mandic. 2001. In *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*. [link].

Sabyasachi Mukhopadhyay Mrityunjoy Panday Deb Prakash Chatterjee, Anirban Mukherjee. 2021. A survey on sentiment analysis.emerging technologies in data mining and information security. In *Proceedings of IEMIS 2020, Volume 2 (pp.259-271)*.

Aysu Ezen-Can. 2020. A comparison of lstm and bert for small corpus. In *arXiv:2009.05451v1 [cs.CL]*.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Kiemute Oyibo Ifeoma Adaji and JulitaVassileva. 2020. E-commerce shopping motivation and the influence of persuasive strategies. In *ORIGINAL RESEARCH, Frontiers in Artificial Intelligence*.

Kenton Lee Kristina Toutanova Jacob Devlin, Ming-Wei Chang. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

D. Jurafsky and J. Martin. 2009. In *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. [link].

Ivan Suzdalev Kornelija Tunaityte Justinas Janulevicius Antanas Cenys Justinas Rastenis, Simona Ramanauskaite. 2021. Multi-language spam/phishing classification by email body text: Toward automated security incident investigation.

Mahmoud Khonji, Youssef Iraqi, and Andrew Jones. 2013. Phishing detection: A literature survey. volume 15, pages 2091–2121.

Adaji Rita Orji Kiemute Oyibo, Ifeoma and JulitaVassileva. 2018. The susceptibility of africans to persuasive strategies: A case study of nigeria. In *Persuasive Technology International Workshop 2018*.

Kamil Halouzka Pavel Kozak Ladislav Burita, Petr Matoulek. 2021. Analysis of phishing emails.

Paul Pu Liang Soujanya Poria Prateek Vij Erik Cambria Louis-Philippe Morency, Amir Zadeh. 2018. Multi-attention recurrent network for human communication comprehension. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

MakuochiNkwo and Rita Orji. 2018. Persuasive technology in african context: Deconstructing persuasive techniques in an african online marketplace. In *2nd African Computer Human Interaction Conference (AfriCHI'18)*.

Erik Cambria Md Shad Akhtar, Asif Ekbal. 2020. How intense are you? predicting intensities of emotions and sentiments using stacked ensemble. In *IEEE Computational Intelligence Magazine*.

Hugo Mercier. 2017. How gullible are we? a review of the evidence from psychology and social science.

Nikolaos Polatidis Merton Lansley and Stelios Kapetanakis. 2019. Seader: A social engineering attack detection method based on natural language processing and artificial neural networks. In *Computational Collective Intelligence*.

Stelios Kapetanakis Merton Lansley, Nikolaos Polatidis. 2019. A social engineering attack detection method based on natural language processing and artificial neural networks.

Tom Mitchell. 1997. In *Machine Learning*. [link].

Sagar Pande Neha Jadav and Aditya Khamparia. 2018. Twitter sentiment analysis using deep learning. In *IOP Conference Series: Materials Science and Engineering*.

Shervin Malmasi Mihaela Vela Liviu P Dinu Josef Van Genabith Octavia-Maria Sulea, Marcos Zampieri. 2017. Exploring the use of text classification in the legal domain. In *arXiv preprint arXiv:1710.09306*.

Chris Cornelis OlhaKaminska and Veronique Hoste. 2021. Fuzzy-rough nearest neighbour approaches for emotion detection in tweets. In *the IJCRS 2021 conference, organized jointly with IFSA-EUSFLAT 2021*.

Jennifer Carter Hamido Fujita Orestes Appel, Francisco Chiclana. 2021. Consensus in sentiment analysis. In *Fuzzy Logic (pp.35-49)*.

Deepak Upadhyay Oza Pranali P. 2020. Review on phishing sites detection techniques.international journal of engineering and technical research v9(04).

Aaron Crowson Christopher Mayhorn Patrick Lawson, Carl Pearson. 2020. Email phishing and signal detection: How persuasion principles and personality influence response patterns and accuracy.

Andrea Desantis Yi-Fang Hsu Lionel Granjon Claire Sergent Florian Waszak Pierre O. Jacquet, Valentin Wyart. 2018. Human susceptibility to social influence and its neural correlates are related to perceived vulnerability to extrinsic morbidity risks.

Bennett Kleinberg Alexandra Lefevre Rada Mihalcea, Veronica Perez-Rosas. 2017. Automatic detection of fake news. In *arXiv:1708.07104v1 [cs.CL]*.

Katia Sycara Rahul Radhakrishnan Iyer. 2019. An unsupervised domain-independent framework for automated detection of persuasion tactics in text.

Susmi Jacob Vinod P Shilpa P C, Rissa Shereen. 2021. Sentiment analysis using deep learning. In *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*.

Akira Yamada Avumu Kubota Shoma Tanaka, Takashi Matsunaka. 2021. Phishing site detection using similarity of website structure. In *IEEE Conference on Dependable and Secure Computing (DSC)*.

Manish Kumar Shubham Khera. 2020. The comparative analysis with bert and elmomethods for movie reviews prediction using nlp.

Sriparna Saha Pushpak Bhattacharyya Shweta Yadav, Asif Ekbal. 2018. Medical sentiment analysis using social media: towards building a patient assisted system. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Natalie DeClerck Sridhar Ramaswamy. 2018. Customer perception analysis using deep learning and nlp complex adaptive systems conference with theme: Cyber physical systems and deep learning.

Pushpak Bhattacharyya Subhabrata Mukherjee. 2012. Feature specific sentiment analysis for product reviews. In *International Conference on Intelligent Text Processing and Computational Linguistics*.

Raksha Sharma Vansh Gupta. 2021. Roberta model with data augmentation for persuasion techniques detection.

Aspect Based Sentiment Analysis Using Spectral Temporal Graph Neural Network

Abir Chakraborty
Microsoft India
Abir.Chakraborty@microsoft

Abstract

The objective of Aspect Based Sentiment Analysis is to capture the sentiment of reviewers associated with different aspects. However, complexity of the review sentences, presence of double negation and specific usage of words found in different domains make it difficult to predict the sentiment accurately and overall a challenging natural language understanding task. While recurrent neural network, attention mechanism and more recently, graph attention based models are prevalent, in this paper we propose graph Fourier transform based network with features created in the spectral domain. While this approach has found considerable success in the forecasting domain, it has not been explored earlier for any natural language processing task. The method relies on creating and learning an underlying graph from the raw data and thereby using the adjacency matrix to shift to the graph Fourier domain. Subsequently, Fourier transform is used to switch to the frequency (spectral) domain where new features are created. These series of transformation proved to be extremely efficient in learning the right representation as we have found that our model achieves the best result on both the SemEval-2014 datasets, i.e., "Laptop" and "Restaurants" domain. Our proposed model also found competitive results on the two other recently proposed datasets from the e-commerce domain.

1 Introduction

With the proliferation of online shopping consumers are relying more and more on ratings, and reviews available from past customers. Aspect-based sentiment analysis (ABSA) tries to understand customers' granular opinion on different dimensions (aspects) of a product which helps in understanding its current limitations and planning for the next round of improvements. These reviews invariably talk about multiple aspects (sometimes in the same sentence) with the presence of mixed feedbacks, positive and negative. The presence of multiple aspects and opposite sentiments makes the task of ABSA challenging and since its birth in 2014 (SemEval-2014 Task-4, Pontiki et al.) the task has seen a variety of different approaches and still enjoys considerable attention from the research community.

The key to an improved performance in ABSA lies in the ability to identify the aspects and the corresponding sentiments with the help of connections between them. A typical example would be "The price is reasonable although the service is poor" where price and service are the aspects with positive and negative sentiments, respectively. It is clear from the sentence that the corresponding modifiers, reasonable and poor, are driving the respective sentiments. However, in another example, "I had the salmon dish and while it was fine, for the price paid, I expected it to have some type of flavor", it is not clear which words or phrases can be identified as responsible for the negative sentiment. Similar examples are "prices are in line" (neutral sentiment) and "For the price, you can not eat this well in Manhattan" (positive sentiment). Thus, while one may hope that capturing the syntactical structure and dependency between words will help in improving the sentiment identification it is also important to understand the meaning and application of words in general setting and not to be gleaned from the limited examples present in the modest ABSA datasets (2328 and 3608 training examples for the popular "Laptop" and "Restaurant" domains, respectively). Recent successes with BERT and RoBERTa based encoders indicate that a powerful word representation in the context of the surrounding words would help in establishing the required connections between the aspects and sentiments.

While large model-based encoders provide a powerful initial representation, subsequent transfor-

Proceedings of the 18th International Conference on Natural Language Processing, pages 508–518
Silchar, India. December 16 - 19, 2021. ©2021 NLP Association of India (NLPAI)

mations possibly involving attention and additional embeddings from parts-of-speech or dependency parser are equally important for the quality of the final prediction. Recent trends also indicate a confluence of deep encoders and graph-based representation of words (nodes) where semantic relations are captured in the graph representations either by converting a dependency graph or by utilizing the nearest neighborhood of words.

Instead of working on a graph structure stemming from the language itself, a different approach of constructing an underlying graph can be by learning it from the encoded representations. This is motivated by recent advances in the forecasting literature where in absence of any obvious underlying graph, a graph is created from encoded representation by applying self-attention. Armed with this new graph subsequent transformations including Graph Fourier Transform (GFT) and Discrete Fourier Transform (DFT) proved to be extremely efficient in understanding the correlation between different dimensions and increasing the overall performance of the model.

Motivated by the success of this approach in the forecasting domain, we apply the same for ABSA although we believe there can be many other applications. There are three key components in this technique, (a) create a graph from the encoded representation using self-attention, (b) use the graph Laplacian to transform from the vertex domain to graph-spectral domain and (c) apply DFT to transform from graph-spectral to frequency domain where convolution operator extracts features. To the best of our knowledge these components approach has not been considered in solving any Natural Language applications and would open numerous possibilities for further modifications and improvements.

The organization of the paper is as follows. In the next section we provide a detailed literature survey on the techniques employed for ABSA. Next, we present the details of the proposed model. Subsequently, the model predictions and comparisons with other baseline methods are discussed. Finally, conclusions are drawn and scope for future works is outlined.

2 Related Work

Since it's introduction in 2014 in Sem-Eval (Task-4), (Pontiki et al.) ABSA has come a long way from initial SVM classifier with handcrafted features to deep learning classifiers based on RNN, Transformer and memory network. We broadly categorize these models into three groups, (a) RNN with attention, (b) memory network and (c) models that use Transformer architecture and/or pre-trained language model. One of the first application of LSTM was proposed by (Wang et al., 2016) where attention mechanism was applied on LSTM output that embedded both the words and aspects individually. (Wang et al., 2018) used a hierarchical network of Bi-directional LSTMs with attention at word and phrase level. A new attention model was proposed by He et al. (2018) that improved the performance of the previous LSTM based models. Ma et al. (2017) proposed an interactive network to learn separate embeddings for the context and target with two sets of LSTMs that attends to specific part of the context based on the target (aspect). Relations between aspects are further investigated by (Hazarika et al., 2018) where hierarchical LSTM structure was used to capture inter-aspect dependency. An attention-over-attention model was proposed by (Huang et al., 2018) that modelled aspects and sentiments together and explicitly captured their interactions. Similar hierarchical attention model was proposed by (Li et al., 2018) which emphasised on the position information of the aspect.

On the models based on memory network, Tang et al. (2016) and Chen et al. (2017) designed deep memory networks to with weighted memory mechanism to capture relations between aspects and sentiments separated by long distance. Tay et al. (2017) introduced dyadic memory network for ABSA where relevant memory information is adaptively used based on the input query. Cheng et al. (2017) proposed hierarchical attention network to have separate aspect attention and sentiment attention that found better matching between previously unseen aspect and sentiment words. Lin et al. (2019) proposed a new mask memory network with semantic dependency that exploited inter-aspect relations for aspects in the same sentence.

With the advent of Transformer (Vaswani et al., 2017) and strong baselines reported for different NLP tasks with BERT (Devlin et al., 2019) based architectures there are quite a few BERT based models for ABSA. Hoang et al. (2019) used sentence-pair classification task to reformulate aspect extraction and aspect polarity classification. Zeng et al. (2019) used BERT embeddings to create a local and global representation of the contexts that were further processed via multi-head self-attention. Xu et al. (2019) created a novel task called Review Reading Comprehension from the ABSA datasets and applied BERT to answer the review questions. Li et al. (2019) used BERT as embedding layer together with CRF for end-to-end ABSA. BERT embeddings are also used by Song et al. (2019) for ABSA with label smoothing regularization. Sun et al. (2019a) constructed ABSA as sentence-pair classification task by constructing auxiliary sentences. Phan and Ogunbona (2020) combined POS embeddings, dependency embeddings and self-attention with RoBERTa (Liu et al., 2019) embeddings to further improve on the aspect classification results.

There are not many applications of graph neural networks for ABSA available in the literature. Zhang et al. (2019) and Sun et al. (2019b) used graph convolution network (GCN) where the graph structure was learnt from the dependency tree. Similarly, Huang and Carley (2019) used graph attention network (GAT) to establish the dependency between words without paying specific attention to the aspects and their opinions. Wang et al. (2020) modified the original dependency tree to create an aspect-oriented dependency tree that was used further in a relational GAT (R-GAT) where different relations contributed differently in the computation of nodal representations.

While the above mentioned approaches rely on a graph structure that emerges naturally from the syntactic structure of the examples it is worth exploring if there is a possibility of learning the graph structure itself from the presented data. This idea is borrowed from Forecasting literature where state-of-the-art models are based on GCN originated from the theory of Graph Fourier Transform (GFT). In addition to GCN and temporal modules like LSTM or GRU, it has also been shown that feature processing in the spectral domain can substantially improve the model performance (Cao et al., 2020). While the application of Fourier transform is not common in the natural language processing (NLP) domain, it has been observed recently (Lee-Thorp et al., 2021) that the self-attention layer in the Transformer can be replaced by a standard Fourier Transform and still achieving 92-97% of the original accuracy.

3 Methodology

The overall architecture of the current method closely follows the architecture of Spectral Temporal Graph Neural Network (STGNN) (Cao et al., 2020) with some minor modifications (see Fig. 1). However, for the sake of completeness the components of STGNN are described here. There are five major transformations that any sentence will be subjected to, (1) encoding by an embedding layer, (2) processing by an RNN (we call it the encoder) and create a graph structure, (3) transformation from vertex domain to graph spectral domain using the eigenvectors of this graph, (4) discrete Fourier transform in the graph spectral domain and (5) filtering by convolution layers in the graph spectral frequency domain. Subsequently, inverse Fourier transform and inverse graph Fourier transform are applied sequentially to bring the representation back to the graph vertex domain. These transformations can be broadly combined into three key components, (1) embedding layer, (2) latent correlation layer (LCL) and (3) spectral block layer (SBL). The details of each layer are given below:

3.1 Embedding Layer

The embedding layer converts a sentence into a sequence of vectors of some suitable dimension (h). Some of the popular choices are (a) Glove vector ($h = 300$), (b) different BERT models (mostly, $h = 768$) or RoBERTa models ($h = 768$). We have used BERT with 768-dimensional output for all subsequent experiments. While there are different ways of representing a sentence (a) representation of the [CLS] token, (b) average of all the tokens at the last layer or (c) representation of the tokens at the last layer. Here we use the last option where we pass the original sentence along with the aspect term separated by a [SEP] token, i.e., [CLS] + sentence + [SEP] + aspect + [SEP], where + indicates concatenation. Two sentence segments are also created to distinguish between the original sentence and the aspect terms.

3.2 Latent Correlation Layer

Given an embedded vector of a sentence LCL learns an underlying graph structure and emits the corresponding graph Laplacian (L). This is where STGNN differs from other methods where the graph is computed from the presented data and does not use any external information. First, the encoded representation from the previous layer ($\hat{X} \in \mathbb{R}^{b\times s\times h}$, where b is the batch size and s is the sequence length) is passed through an RNN (of hidden dimension h, we have experimented with GRU, LSTM and Bi-LSTM) where the input is posed as a sequence of h elements of dimension s.

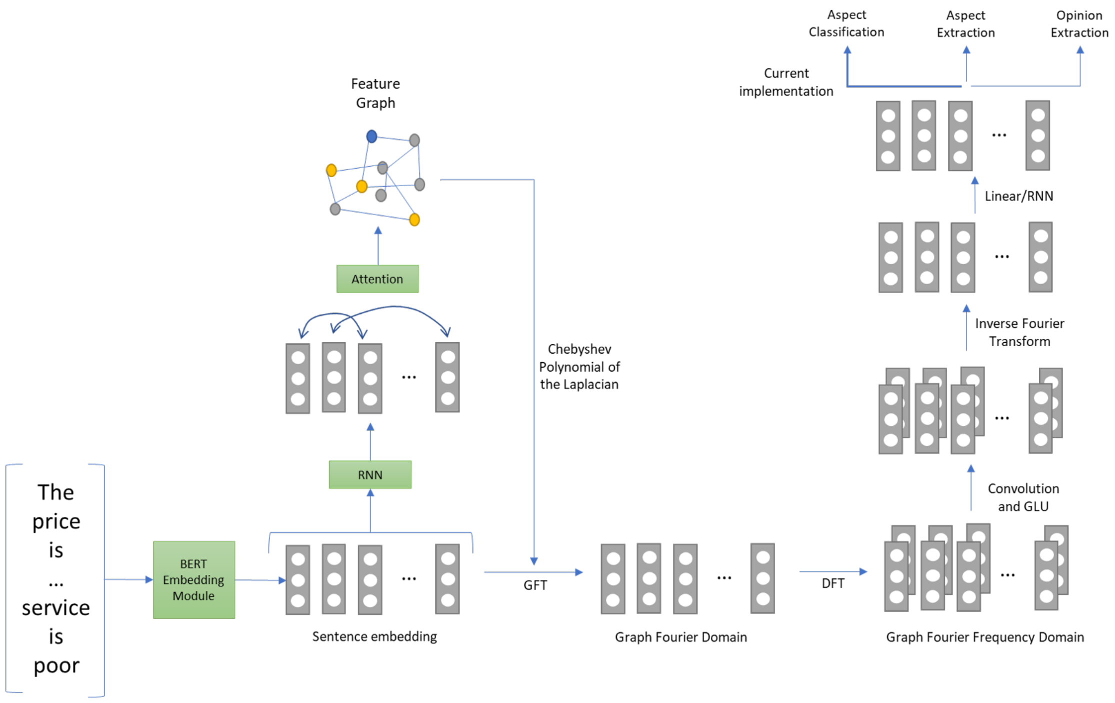


Figure 1: Spectral-Temporal Graph Neural Network model for aspect polarity detection. An example sentence is embedded (using e.g., BERT) first and passed through an RNN and self-attention layer to create a feature graph. The Laplacian of this graph and DFT are used successively to convert the sentence representation into the graph Fourier frequency domain. After extracting features in this domain using 1D convolution and GLU, they are subsequently transformed back to the original domain using inverse Graph Fourier transform and inverse DFT. The final representation is connected to the logits through a fully-connected layer.

The last hidden state of the RNN ($\breve{X} \in \mathbb{R}^{b\times h\times h}$) is taken as a representation of the entire sentence and passed through an attention layer:

$$W = Softmax\frac{(QK^T}{\sqrt{d}}),\ Q = \breve{X}W^Q,\ K = \breve{X}W^K \tag{1}$$

where Q and K can be thought as the query and key learnt from the RNN output through trainable weights W^Q and W^K. The output matrix $W \in \mathbb{R}^{h\times h}$ is taken as the weighted adjacency matrix of the graph. The adjacency matrix is further processed to create the Laplacian matrix defined as $\mathcal{L} = I_h - D^{-1/2}WD^{1/2}$ where I_h is the h-dimensional identity matrix, D is the diagonal degree matrix with $D_{ii} = \sum_j W_{ij}$. The eigenvectors of the Laplacian, U (where $\mathcal{L} = U\Lambda U^T$), is used for GFT defined as $GFT(X) = \bar{X} = U^TX$ and inverse-GFT becomes $X = U\bar{X}$. While backpropagation can be applied through eigenvalue decomposition it is often numerically unstable (Wang et al., 2019). Instead, we apply Chebyshev polynomial approximation (Shuman et al., 2011) which only requires Chebyshev polynomials of the Laplacian (L) up to a specified order. Thus, if the order of the Chebyshev polynomial considered is k then the GFT is defined as

$$\bar{X} = [T_0(\mathcal{L}), T_1(\mathcal{L}), \ldots, T_{k-1}(\mathcal{L})]\,X, \tag{2}$$

where $T_\ell(\mathcal{L}) \in \mathbb{R}^{N\times N}$.

3.3 Spectral Block Layer

The transformed representation in the graph Fourier domain is further transformed into frequency domain using DFT. Subsequently 1D convolution followed by a Gated Linear Unit (Dauphin et al., 2017) (originally applied for language modelling) is applied to both the real and imaginary components independently to extract novel features. The output of GLU is transformed back to the time domain using inverse Fourier transform. Subsequently, a linear transformation (akin to inverse GFT) is applied to map back to the vertex domain. Specifically, the DFT output has real and imaginary components, $\hat{X}^r$ and $\hat{X}^i$, that are processed by the same operators (but different parameters) in parallel. The operation can be written as

$$M^r(\hat{X}^r) = GLU\left(\theta^r(\hat{X}^r), \theta^r(\hat{X}^r)\right) \tag{3}$$

Dataset	Positive		Neutral		Negative		Total	
	Train	Test	Train	Test	Train	Test	Train	Test
Laptop	994	341	464	169	870	128	2328	638
Restaurants	2164	728	637	196	807	196	3608	1120
Men's Tshirt	1122	270	50	16	699	186	1871	472
Television	2540	618	287	67	919	257	3746	942

Table 1: Statistics of the datasets used in this work

where θ^r is the convolution kernel of size 3, σ denotes sigmoid function, $GLU(x, x) = x \odot \sigma(x)$ and $\odot$ is the element-wise Hadamard product. The same operation is applied to the imaginary components and they are combined together as $M^r(\hat{X}^r) + jM^i(\hat{X}^i)$ $(j^2 = -1)$ before applying inverse DFT.

The combined transformation of the LCL and SBL can be thought as another layer that generates an output of dimension same as that of the input which is very similar to the operation of the Transformer layer (Vaswani et al., 2017), i.e., the output $\tilde{X} \in \mathbb{R}^{b \times s \times h}$. This processed version of the original input X can be transformed further depending upon the nature of the task. For aspect polarity, we explore different options like (1) two fully-connected (FC) layers, (2) GRU followed by a FC layer and (3) LSTM followed by a FC layer. For all these cases, the second FC layer always has two sub-layers with a leaky Relu transfer function in between. The second sub-layer emits raw score of dimension three corresponding to the three sentiment classes (positive, neutral and negative). We use categorical cross-entropy loss with L_2 regularization. The overall time complexity of self-attention and GFT is $O(h^3)$, where h is both the BERT embedding dimension and the hidden dimension of the encoder (GRU/LSTM). The complexity of DFT is $slog(s)$ where s is the sequence length.

4 Experiments

In this section, we first describe the datasets used for the evaluation of our proposed method and the other baseline methods employed for comparison. Then, we report the experimental results conducted from different perspectives. Finally, error analysis and discussion are conducted with a few representative examples.

4.1 Datasets

We use four public sentiment analysis datasets, two of them are the commonly used Laptop and the Restaurant review datasets from SemEVal-14 task (Pontiki et al., 2014) and other two are recently released and based on e-commerce reviews, namely, Men's T-shirt and Television ((Mukherjee et al., 2021)). Statistics of these datasets are given in Table 1. Looking at the datasets it is apparent that in general we do not have enough training data for most of the deep learning based models and one has to be careful to avoid over-fitting. We also experiment on the "hard-data" as defined by (Xue and Li, 2018) where examples with multiple aspects and different polarities are identified. All experiments were conducted on Tesla K-80 with 12 GB GPU.

4.2 Implementation Details

We extend the codebase of (Mukherjee et al., 2021) by adding our proposed model. We have used 768-dimensional embeddings of BERT (Devlin et al., 2019) implemented in the PyTorch environment. There are several hyperparameters that we should tune for, namely, learning rate, dropout rate, regularization parameter L_2 weights and STGNN specific parameters like the number of layers, encoder and decoder types (fully-connected, GRU, LSTM, Bi-LSTM etc.) etc. However, what we have found is that the optimal set of parameters can be different for different datasets and it would take substantial amount of computational effort to obtain all four of them.

In this work we have not done an extensive search of the hyper-parameter space. Instead, we started with the baseline parameters used earlier (Mukherjee et al., 2021) and modified only the L_2 weight that we found to be significantly affecting the test results. Thus, all subsequent results are based on $L_2 = 2 \times 10^{-7}$ whereas the other parameters are as follows: (a) learning rate $= 1 \times 10^{-5}$, (b) dropout = 0 and (c) batch size $= 32$. We have used Adam optimizer with the default parameters

Model	Reported (no held out)		Reproduced (no held out)		Reproduced using 15% held out	
	Accuracy	F1	Accuracy	F1	Accuracy	F1
ATAE-LSTM	68.70	-	60.28	44.33	58.62 (33.47)	43.27 (29.01)
RAM	74.49	71.35	72.82	68.34	70.97 (56.04)	65.31 (55.81)
IAN	72.10	-	69.94	62.84	69.40 (48.91)	61.98 (48.75)
BERT-SPC	78.99	75.03	78.72	74.52	77.24 (59.21)	72.80 (59.44)
BERT-AEN	79.93	76.31	78.65	74.26	75.71 (46.53)	70.02 (45.22)
LCF-BERT	77.31	75.58	**79.75**	**76.10**	77.27 (62.57)	72.86 (62.71)
R-GAT+BERT	78.21	74.07	79.15	75.14	75.64	69.52
STGNN-GRU	-	-	79.09*	75.28*	**78.72**(**64.36**)	**74.84**(**64.34**)

Table 2: Comparison of predictions on the Laptop dataset. The best results are highlighted in bold and * next to a number indicates the second best result.

($\beta_1 = 0.9$ and $\beta_2 = 0.999$) and weight decay. As reported by (Mukherjee et al., 2021) most of the earlier studies did not set aside a separate test set and the same dataset was used for validation. However, in this work we follow the same process of keeping 10-15% of the train data as the validation set. The first pass runs over all the epochs and the optimal epoch number is noted that corresponds to the maximum validation accuracy. Next, the entire training set is considered for training but only up to the optimal epoch and finally the model performance on the test data is reported.

4.3 Baseline Methods

We compare with the methods studied by (Mukherjee et al., 2021) along with the R-GAT model of (Wang et al., 2020). The methods compared by (Mukherjee et al., 2021) can be broadly categorized into two classes, (a) memory network based and (b) BERT based. While memory network based models have fewer parameters and better suited for the small datasets the BERT based methods are dominating the ABSA landscape and their success can be attributed to the huge pre-training corpora that helps in better understanding of words and their associations. A brief description of the methods considered here are given below:

1. ATAE-LSTM (Wang et al., 2016) where separate embeddings are used for the aspects and concatenated with word embeddings followed by an attention layer.

2. Recurrent Attention on Memory (RAM, (Chen et al., 2017)) where memory network is used to capture relations between aspects and sentiments separated by long distance.

3. Interactive Attention Network (IAN, (Ma et al., 2017)) where two sets of LSTMs are used to learn the embeddings of the context words and target (aspect). The attention based representations are then concatenated to predict the aspect polarity.

4. BERT-SPC, which is a baseline BERT model that treats sentiment classification as a sentence pair classification task where the pooled output of a modified sentence $[CLS]+$ context + $[SEP]+$ target + $[SEP]$ is passed to a fully-connected layer.

5. BERT-AEN (Song et al., 2019) that uses attentional encoder network with label smoothing regularization.

6. The local context focus BERT (LCF-BERT, (Zeng et al., 2019) where a local and global representation of the contexts are created through BERT that are further processed via multi-head self-attention.

In addition, we also consider the R-GAT model that combines the power of BERT with Graph attention network and reported the best result so far for both the Laptop and Restaurants domain.

5 Results & Analysis

For all the baseline models, it is difficult to know the exact hyperparameter settings in order to reproduce the results. Instead, we relied on the results that are obtained by (Mukherjee et al., 2021). We have also included the originally reported results for the sake of completion and easy comparison. For all the datasets we have two sets of results, (a) the test set is used as a validation set and the model is decided based on the epoch with the best test set accuracy; and (b) 15% of the training data is used as a validation set that decides the optimum number

Model	Reported (no held out)		Reproduced (no held out)		Reproduced using 15% held out	
	Accuracy	F1	Accuracy	F1	Accuracy	F1
ATAE-LSTM	77.20	-	73.71	55.87	73.29 (52.41)	54.59 (47.35)
RAM	80.23	70.80	78.21	65.94	76.36 (59.29)	63.15 (56.36)
IAN	78.60	-	76.80	64.24	76.52 (57.05)	63.84 (55.11)
BERT-SPC	84.46	76.98	85.04*	78.02*	**84.23** (68.84)	76.28 (68.11)
BERT-AEN	83.12	73.76	81.73	71.24	80.07 (51.70)	69.80 (48.97)
LCF-BERT	87.14	81.74	**85.94**	**78.97**	84.20* (69.38)	76.28 (69.64)
R-GAT+BERT	86.60	81.35	85.27	78.40	83.40	75.74
STGNN-GRU	-	-	84.93	77.65	83.66 (69.29)	75.33 (68.45)
STGNN-LSTM	-	-	-	-	84.20* (**70.98**)	**76.55** (**70.44**)

Table 3: Comparison of predictions on the Restaurants dataset. The best results are highlighted in bold and * next to a number indicates the second best result.

of epochs. Subsequently, the model is trained on the full train set till the optimum number of epochs and results are reported on the test set. For both the cases, average scores over 5 runs are reported for all the experiments.

5.1 Model Performance

Table 2 presents the results from the baseline models as well as our current model for the Laptop dataset. The first two columns show the originally reported test accuracy and F1-score without any held out validation data. The next two columns show the same metrics as obtained by (Mukherjee et al., 2021) again without any separate validation data. The last two columns show the same metrics with 15% validation data (created from train set). As we can see the current method obtains the best result for both the accuracy and F1-score for this setup with a substantial improvement over the next best result from LCF-BERT. Our model also works well on the hard dataset with an improvement of 1.79 and 1.63 percent point, respectively, for the accuracy and F1-score.

On the Restaurant dataset (Table 3) we show two different predictions from our model, one with GRU encoder and the second one with LSTM encoder. For GRU encoder, our model predictions are close to the best predictions of BERT-SPC and LCF-BERT while the gap in accuracy on the hard dataset is minimal. It is to be noted that the same set of hyperparameters is used in this case and not tuned specifically for the Restaurant dataset. Similar trend is also observed for the BERT-AEN and R-GAT models where the performance on the Laptop dataset is significantly better compared to the Restaurant dataset. Using LSTM encoder, on the other hand, our model accuracy on the whole dataset is same as that of the best model whereas, on the hard dataset STGNN prediction outperforms the current best model. In case of F1 score, our model outperforms both on the overall and hard dataset. It is to be noted that on the Laptop dataset, the LSTM encoder based STGNN model does not perform better than the GRU based model. More on the choice of encoder is discussed later.

For the Men's T-Shirt and Television dataset all the previous results are reported by (Mukherjee et al., 2021). Table 4 shows the comparison for the Men's T-Shirt dataset where our model achieves the best results for the no held out scenario. For the 15% validation data based case, the present model achieves competitive performance on the complete test data (a gap of only 0.2 percent point on accuracy). However, the gap increases to 1.67 percent point on the hard dataset. It is to be noted that there are only 48 examples in the hard test set. Similarly, on the Television dataset (shown in Table 5) our model achieves comparable results for both no held out and 15% held out data. For the first case (no separate validation set) the gap in accuracy and F1-score with the best performing model (LCF-BERT) is 0.63 and 0.28 percent point, respectively. For the 15% held out data, our model achieves the second best results with a gap of 0.21 and 0.56 percent points, respectively, on the accuracy and F1-score. Similarly, on the hard slice the gaps are also minimal at 0.4 percent point.

5.2 Error Analysis

We have also conducted a detailed analysis of the errors made by our model to understand if any discernible pattern exists. A summary of the dis-

Model	no held out		using 15% held out	
	Accuracy	F1	Accuracy	F1
ATAE-LSTM	83.13	55.98	81.65 (58.33)	54.84 (39.25)
RAM	90.51	61.93	88.26 (83.33)	59.67 (56.01)
IAN	87.58	59.16	87.41 (63.75)	58.97 (42.85)
BERT-SPC	93.13	73.86	92.42 (89.58)	**73.83** (60.62)
BERT-AEN	88.69	72.25	87.54 (50.42)	59.14 (32.96)
LCF-BERT	93.35	72.19	91.99 (91.67)	72.13* (62.30)
STGNN-GRU	**93.60**	**78.77**	92.21* (90.0*)	71.09 (60.90)

Table 4: Comparison of predictions on the Men's T-Shirt dataset. The best results are highlighted in bold and * next to a number indicates the second best result.

Model	no held out		using 15% held out	
	Accuracy	F1	Accuracy	F1
ATAE-LSTM	81.10	53.71	79.68 (53.92)	52.78 (39.13)
RAM	84.29	58.68	83.02 (64.31)	58.50 (50.07)
IAN	82.42	57.15	80.49 (54.31)	56.78 (41.67)
BERT-SPC	89.96*	74.68	88.56 (80.20)	74.81 (**74.32**)
BERT-AEN	87.09	67.92	85.94 (50.39)	65.65 (38.08)
LCF-BERT	**90.36**	**76.01**	**90.00**(**80.98**)	**75.86** (73.72*)
STGNN-GRU	89.73	75.73*	89.79* (80.59*)	75.30* (73.32)

Table 5: Comparison of predictions on the Television dataset.

tribution of the true class for different datasets are provided in Table 6. It can be seen that most of the error is concentrated around the neutral class for the Laptop, Restaurant and Television dataset, whereas, for the Men's T-Shirt dataset the errors are uniform amongst the classes.

For the neutral classes the errors are broadly categorized into two classes:

- **Presence of negation words**, examples: (a) ”which it did not have , only 3 usb 2 ports .”, (b) ”no startup disk was not included but that may be my fault”, (c) ”there is no ””tools”” menu .”, or (d) ”the happy hour is so cheap , but that does not reflect the service or the atmosphere .”

- **Presence of negative/positive adjectives**, examples: (a) the only solution is to turn the brightness down, (b) ”a lot of features and shortcuts on the mbp that i was never exposed to on a normal pc”, (c) ”premium price for the os more than anything else”, or (d) ”tiny restaurant with very fast service .”

while for the positive or negative true classes there are examples of general lack of understanding of the meaning due to their complexity or presence of double negation:

Dataset	Positive	Negative	Neutral
Laptop	32%	15%	53%
Restaurants	25%	20%	55%
Men's Tshirt	30%	36%	34%
Television	33%	23%	43%

Table 6: Distribution of mis-prediction across different true classes

- **Complicated**: (a) ”if you ask me , for this price it should be included”, (b) ”logic board utterly fried , cried , and laid down and died”, (c) ”however , i can refute that osx is “ fast ” .”, or (d) ”the sangria 's - watered down”

- **Double negation**: (a) screen - although some people might complain about low res which i think is ridiculous ., (b) i would have given it 5 starts was it not for the fact that it had windows 8 etc.

In absence of enough training examples the onus of understanding the nuances of the language falls on the word/sentence representation, which also explains the relatively higher success rate of BERT.

6 Conclusion

We present a novel application of graph Fourier transform with spectral feature engineering hitherto limited to forecasting domain. The model learns an underlying graph structure from the raw data created by a BERT encoder. The advantage of this approach is that it does not require dependency parser based graph creation and thereby does not inherit any limitation of the parser. It is shown that the series of transformations involving GFT, DFT, convolution and GLU create powerful representations of the text resulting in the superior performance on SemEval-2014 datasets, namely "Laptop" and "Restaurants" domain. On the "Laptop" dataset we achieved the best results while on the "Restaurants" dataset our performance is at par with the current best prediction. On the recently released e-commerce datasets, our model performance is very competitive with a gap of 0.2-0.4 percent points. Although we have not done a full-scale hyper-parameter tuning, the effect of different components like the initial encoder and the final layer is studied. It is observed that the same set of hyper-parameters and architecture will not generate the best result across all the datasets.

There are several possible future directions of work. If we view the current model as a spectral graph transformer that takes sequential input and generates sequential output there could be several other applications like, sequence tagging or natural language generation. Also, we have evaluated only BERT for sentence encoding and in future, other language models like RoBERTa and GPT can be explored.

References

Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Congrui Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. 2020. Spectral temporal graph neural network for multivariate time-series forecasting. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.*

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Recurrent neural network-based sentence encoder with gated attention for natural language inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP*, pages 36–40, Copenhagen, Denmark. Association for Computational Linguistics.

Jiajun Cheng, Shenglin Zhao, Jiani Zhang, Irwin King, Xin Zhang, and Hui Wang. 2017. *Aspect-Level Sentiment Classification with HEAT (HiErarchical ATtention) Network*, page 97–106. Association for Computing Machinery, New York, NY, USA.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 933–941. PMLR.

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.

Devamanyu Hazarika, Soujanya Poria, Prateek Vij, Gangeshwar Krishnamurthy, Erik Cambria, and Roger Zimmermann. 2018. Modeling inter-aspect dependencies for aspect-based sentiment analysis. 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 266–270, New Orleans, Louisiana. Association for Computational Linguistics.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018. Effective attention modeling for aspect-level sentiment classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1121–1131, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Mickel Hoang, Oskar Alija Bihorac, and Jacobo Rouces. 2019. Aspect-based sentiment analysis using BERT. In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pages 187–196, Turku, Finland. Linköping University Electronic Press.

Binxuan Huang and Kathleen M. Carley. 2019. Syntax-aware aspect level sentiment classification with graph attention networks.

Binxuan Huang, Yanglan Ou, and Kathleen M. Carley. 2018. Aspect level sentiment classification with attention-over-attention neural networks.

James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. 2021. Fnet: Mixing tokens with fourier transforms.

Lishuang Li, Yang Liu, and AnQiao Zhou. 2018. Hierarchical attention based position-aware network for aspect-level sentiment analysis. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 181–189, Brussels, Belgium. Association for Computational Linguistics.

Xin Li, Lidong Bing, Wenxuan Zhang, and Wai Lam. 2019. Exploiting BERT for end-to-end aspect-based sentiment analysis. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 34–41, Hong Kong, China. Association for Computational Linguistics.

Peiqin Lin, Meng Yang, and Jianhuang Lai. 2019. Deep mask memory network with semantic dependency and context moment for aspect level sentiment classification. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5088–5094. International Joint Conferences on Artificial Intelligence Organization.

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.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4068–4074.

Rajdeep Mukherjee, Shreyas Shetty, Subrata Chattopadhyay, Subhadeep Maji, Samik Datta, and Pawan Goyal. 2021. Reproducibility, replicability and beyond: Assessing production readiness of aspect based sentiment analysis in the wild. In *Advances in Information Retrieval*, pages 92–106, Cham. Springer International Publishing.

Minh Hieu Phan and Philip O. Ogunbona. 2020. Modelling context and syntactical features for aspect-based sentiment analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3211–3220, Online. Association for Computational Linguistics.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. SemEval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35, Dublin, Ireland. Association for Computational Linguistics.

David I Shuman, Pierre Vandergheynst, and Pascal Frossard. 2011. Chebyshev polynomial approximation for distributed signal processing. In *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pages 1–8.

Youwei Song, Jiahai Wang, Tao Jiang, Zhiyue Liu, and Yanghui Rao. 2019. Targeted sentiment classification with attentional encoder network. *Lecture Notes in Computer Science*, page 93–103.

Chi Sun, Luyao Huang, and Xipeng Qiu. 2019a. Utilizing bert for aspect-based sentiment analysis via constructing auxiliary sentence.

Kai Sun, Richong Zhang, Samuel Mensah, Yongyi Mao, and Xudong Liu. 2019b. Aspect-level sentiment analysis via convolution over dependency tree. 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 5679–5688, Hong Kong, China. Association for Computational Linguistics.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 214–224, Austin, Texas. Association for Computational Linguistics.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2017. Dyadic memory networks for aspect-based sentiment analysis. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, page 107–116, New York, NY, USA. Association for Computing Machinery.

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

Jingjing Wang, Jie Li, Shoushan Li, Yangyang Kang, Min Zhang, Luo Si, and Guodong Zhou. 2018. Aspect sentiment classification with both word-level and clause-level attention networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4439–4445. International Joint Conferences on Artificial Intelligence Organization.

Kai Wang, Weizhou Shen, Yunyi Yang, Xiaojun Quan, and Rui Wang. 2020. Relational graph attention network for aspect-based sentiment analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3229–3238, Online. Association for Computational Linguistics.

Wei Wang, Zheng Dang, Yinlin Hu, Pascal Fua, and Mathieu Salzmann. 2019. Backpropagation-friendly eigendecomposition. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

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, Austin, Texas. Association for Computational Linguistics.

Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. 2019. Bert post-training for review reading comprehension and aspect-based sentiment analysis.

Wei Xue and Tao Li. 2018. Aspect based sentiment analysis with gated convolutional networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2514–2523, Melbourne, Australia. Association for Computational Linguistics.

Biqing Zeng, Heng Yang, Ruyang Xu, Wu Zhou, and Xuli Han. 2019. Lcf: A local context focus mechanism for aspect-based sentiment classification. *Applied Sciences*, 9(16).

Chen Zhang, Qiuchi Li, and Dawei Song. 2019. Aspect-based sentiment classification with aspect-specific graph convolutional networks. 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 4568–4578, Hong Kong, China. Association for Computational Linguistics.

Using Random Perturbations to Mitigate Adversarial Attacks on Sentiment Analysis Models

Abigail Swenor and **Jugal Kalita**
University of Colorado Colorado Springs
1420 Austin Bluffs Pkwy
Colorado Springs, CO 80918
aswenor,jkalita@uccs.edu

Abstract

Attacks on deep learning models are often difficult to identify and therefore are difficult to protect against. This problem is exacerbated by the use of public datasets that typically are not manually inspected before use. In this paper, we offer a solution to this vulnerability by using, during testing, random perturbations such as spelling correction if necessary, substitution by random synonym, or simply dropping the word. These perturbations are applied to random words in random sentences to defend NLP models against adversarial attacks. Our Random Perturbations Defense and Increased Randomness Defense methods are successful in returning attacked models to similar accuracy of models before attacks. The original accuracy of the model used in this work is 80% for sentiment classification. After undergoing attacks, the accuracy drops to accuracy between 0% and 44%. After applying our defense methods, the accuracy of the model is returned to the original accuracy within statistical significance.

1 Introduction

Deep learning models have excelled in solving difficult problems in machine learning, including Natural Language Processing (NLP) tasks like text classification (Zhang et al., 2015; Kim, 2014) and language understanding (Devlin et al., 2019). However, research has discovered that inputs can be modified to cause trained deep learning models to produce incorrect results and predictions (Szegedy et al., 2014). Models in computer vision are vulnerable to these attacks (Goodfellow et al., 2015), and studies have found that models in the NLP domain are also vulnerable (Kuleshov et al., 2018; Gao et al., 2018; Garg and Ramakrishnan, 2020). One use of these adversarial attacks is to test and verify the robustness of NLP models.

With the potential for adversarial attacks, there comes the need for prevention and protection. There are three main categories of defense methods: identification, reconstruction, and prevention (Goldblum et al., 2020). Identification methods rely on detecting either poisoned data or the poisoned model (Chen et al., 2019). While reconstruction methods actively work to repair the model after training (Zhu et al., 2020), prevention methods rely on input preprocessing, majority voting, and other techniques to mitigate adversarial attacks (Goldblum et al., 2020; Alshemali and Kalita, 2020). Although most NLP adversarial attacks are easily detectable, some new forms of adversarial attacks have become more difficult to detect like concealed data poisoning attacks (Wallace et al., 2021) and backdoor attacks (Chen et al., 2021). The use of these concealed and hard-to-detect attacks has revealed new vulnerabilities in NLP models. Considering the increasing difficulty in detecting attacks, a more prudent approach would be to work on neutralizing the effect of potential attacks rather than solely relying on detection. Here we offer a novel and highly effective defense solution that preprocesses inputs by random perturbations to mitigate potential hard-to-detect attacks.

2 Related Work

The work in this paper relates to the attack on NLP models using the TextAttack library (Morris et al., 2020), the current state-of-the-art defense methods for NLP models, and using randomness against adversarial attacks.

The TextAttack library and the associated GitHub repository (Morris et al., 2020) represent current efforts to centralize attack and data augmentation methods for the NLP community. The library supports attack creation through the use of four components: a goal function, a search method, a transformation, and constraints. An attack method uses these components to perturb the input to fulfill the given goal function while complying with the constraints and the search method finds transformations that produce adversarial examples. The library contains a total of 16 attack model recipes based on literature. The work reported in this paper pertains to the 14 ready-to-use classification attack recipes from the TextAttack library. We believe that successful defense against such attacks will provide guidelines for the general defense of deep learning NLP classification models.

There are many methods to defend NLP models against adversarial attacks, including input preprocessing. Input preprocessing defenses require inserting a step between the input and the given model that aims to mitigate any potential attacks. Alshemali and Kalita (2020) use an input preprocessing defense that employs synonym set averages and majority voting to mitigate synonym substitution attacks. Their method is deployed before the input is run through a trained model. Another defense against synonym substitution attacks, Random Substitution Encoding (RSE) encodes randomly selected synonyms to train a robust deep neural network (Wang and Wang, 2020). The RSE defense occurs between the input and the embedding layer.

Randomness has been deployed in computer vision defense methods against adversarial attacks. Levine and Feizi (2020) use random ablations to defend against adversarial attacks on computer vision classification models. Their defense is based on a random-smoothing technique that creates certifiably robust classification. Levine and Feizi defend against sparse adversarial attacks that perturb a small number of features in the input images. They found their random ablation defense method to produce certifiably robust results on the MNIST, CIFAR-10, and ImageNet datasets.

3 Input Perturbation Approach & Adversarial Defense

The use and availability of successful adversarial attack methods reveal the need for defense methods that do not rely on detection and leverage intuitions gathered from popular attack methods to protect NLP models. In particular, we present a simple but highly effective defense against attacks on deep learning models that perform sentiment analysis.

The approach taken is based on certain assumptions about the sentiment analysis task. Given a short piece of text, we believe that a human does not need to necessarily analyze every sentence carefully to get a grasp on the sentiment. Our hypothesis is that humans can ascertain the expressed sentiment in a text by paying attention to a few key sentences while ignoring or skimming over the others. This thought experiment led us to make intermediate classifications on individual sentences of a review in the IMDB dataset and then combining the results for a collective final decision.

This process was refined further by considering how attackers actually perturb data. Usually, they select a small number of characters or tokens within the original data to perturb. To mitigate those perturbations, we choose to perform our own random perturbations. Because the attacking perturbations could occur anywhere within the original data, and we do not necessarily know where they are, it is prudent to randomly select tokens for us to perturb. This randomization has the potential to negate the effect the attacking perturbations have on the overall sentiment analysis.

We wish to highlight the importance of randomness in our approach and in possible future approaches for defenses against adversarial attacks. Positive impact of randomness in classification tasks with featured datasets can be found in work using Random Forests (Breiman, 2001). Random Forests have been useful in many domains to make predictions, including disease prediction (Lebedev et al., 2014; Corradi et al., 2018; Paul et al., 2017; Khalilia et al., 2011) and stock market price prediction (kha, 2019; Ballings et al., 2015; Nti et al., 2019). The use of randomness has made these methods of prediction robust and useful. We have chosen to harness the capability of randomness in defense of adversarial attacks in NLP. We demonstrate that the impact randomness has on our defense method is highly positive and its use in defense against adversarial attacks of neural networks should be explored further. We present two algorithms below—first with two levels of randomness, and the second with three.

3.1 Random Perturbations Defense

Our algorithm is based on random processes: the randomization of perturbations of the sentences of a review R followed by majority voting to decide the final prediction for sentiment analysis. We consider each review R to be represented as a set R = $\{r_1, r_2, ..., r_i, ..., r_N\}$ of sentences r_i. Once R

is broken down into its sentences (Line 1 of Algorithm 1), we create l replicates of sentence r_i: $\{\hat{r}_{i1}, ..., \hat{r}_{ij}, ..., \hat{r}_{il}\}$. Each replicate $\hat{r}_{ij}$ has k number of perturbations made to it. Each perturbation is determined randomly (Lines 4-7).

In Line 5, a random token t where $t \in \hat{r}_{ij}$ is selected, and in Line 6, a random perturbation is performed on t. This random perturbation could be a spellcheck with correction if necessary, a synonym substitution, or dropping the word. These perturbations were selected as they are likely to be the same operations an attacker performs, and they may potentially even counter the effect of a large portion of perturbations in attacked data. A spellcheck is performed using SpellChecker which is based in Pure Python Spell Checking. If a spellcheck is performed on a token without spelling error, then the token will not be changed. The synonym substitution is also performed in a random manner. A synonym set for token t is found using the WordNet synsets (Fellbaum, 1998). Once a synonym set is found, it is processed to remove any duplicate synonyms or copies of token t. Once the synonym set is processed, a random synonym from the set is chosen to replace token t in $\hat{r}_{ij}$. A drop word is when the randomly selected token t is removed from the replicate altogether and replaced with a space. Conceptually speaking, the random perturbations may be chosen from an extended set of allowed changes.

Once l replicates have been created for the given sentence r_i and perturbations made to tokens, they are put together to create replicate review set $\hat{R}$ (Line 8). Then, in Line 9, each $\hat{r}_{ij} \in \hat{R}$ is classified individually as $f(\hat{r}_{ij})$ using classifier $f()$. After each replicate has been classified, we perform majority voting with function $V()$. We call the final prediction that this majority voting results in as $\hat{f}(R)$. This function can be thought of as follows (Line 12):

$$\hat{f}(R) = V(\{f(\hat{r}_{ij}) \mid \hat{r}_{ij} \in \hat{R}\}).$$

The goal is to maximize the probability that $\hat{f}(R) = f(R)$ where $f(R)$ is the classification of the original review R. In this paper, this maximization is done through tuning of the parameters l and k. The certainty T for $\hat{f}(R)$ is also determined for each calculation of $\hat{f}(R)$. The certainty represents how sure the algorithm is of the final prediction it has made. In general, the certainty T is determined as follows (Lines 13-17):

$$T = count(f(\hat{r}_{ij}) == \hat{f}(R)) \;/\; N * l.$$

The full visual representation of this algorithm can be seen in Algorithm 1 and in Figure 1.

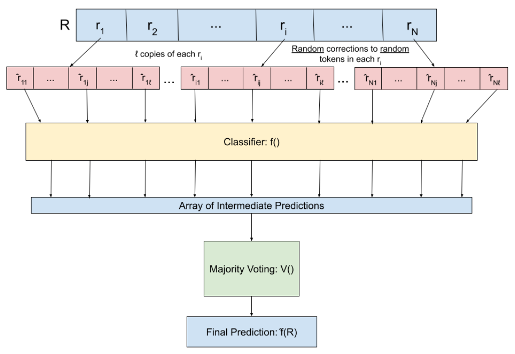


Figure 1: Visual representation of Algorithm 1.

3.2 Increasing Randomness

Our first algorithm represented in Algorithm 1 and in Figure 1 shows randomness in two key points in the decision making process for making the perturbations. This is the main source of randomness for our first algorithm. In our next algorithm, we introduce more randomness into our ideas from our original algorithm to create a modified algorithm. This more random algorithm is visually represented in Figure 2 and presented in Algorithm 2. This new defense method adds a third random process before making random corrections to a sentence. Randomly chosen r_i from R are randomly corrected to create replicate $\hat{r}_j$ which is placed in $\hat{R}$ (Lines 2-6). The original sentence r_i is placed back into R and a new sentence is randomly selected; this is random selection with replacement. This process of random selection is repeated until there is a total of k replicates $\hat{r}_j$ in $\hat{R}$. This algorithm follows the spirit of Random Forests more closely than the first algorithm.

In Line 2, we randomly select a sentence r_i from R. This is one of the main differences between Algorithm 1 for Random Perturbations Defense and Algorithm 2 for Increased Randomness Defense. That extra random element allows for more randomization in the corrections we make to create replicates $\hat{r}_j$. In Lines 3 and 4, the process is practically identical to Lines 5 and 6 in Algorithm 1. The only difference is that only one random correction is being made to get the final replicate $\hat{r}_j$

Algorithm 1: Random Perturbation Defense

```
Result: $\hat{f}(R)$, the classification of $R$ after defense
Input      : Review $R = \{r_1, r_2, ..., r_N\}$ where $r_i$ is a sentence
Parameters : $l$ = number of copies made of each $r$, $k$ = number of corrections made per $r_i$, $C = \{c_1, c_2, ..., c_k\}$, set of corrections
1  $\hat{R} = \emptyset$
2  for $r_i \epsilon R$ do
3      for j = 1 to l do
4          $\hat{r}_{ij} = r_i$
5          for k do
6              Select random token $t$ where $t \epsilon \hat{r}_{ij}$
7              Perform random correction $c \epsilon C$ to $t$
8          end
9          Append $\hat{r}_{ij}$ to $\hat{R}$
10         Classify: $f(r_{ij})$
11     end
12 end
13 $\hat{f}(R) = V(\{f(\hat{r}_{ij}) \mid \hat{r}_{ij} \epsilon \hat{R}\})$, $V()$ is a voting function
14 if $f(\hat{R}) == negative$ then
15     $T = count(f(\hat{r}_{ij}) == negative) / N * l$
16 else
17     $T = count(f(\hat{r}_{ij} == positive) / N * l$
18 end
```

Algorithm 2: Increased Randomness Defense

```
Result: $\hat{f}(R)$, the classification of $R$ after defense
Input      : Review $R = \{r_1, r_2, ..., r_N\}$ where $r_i$ is a sentence
Parameters : $k$ = number of replicates $\hat{r}_j$ made for $\hat{R}$, $C = \{c_1, c_2, ..., c_k\}$, set of corrections
1  $\hat{R} = \emptyset$, $P = []$
2  for j = 1 to k do
3      Randomly select $r_i \in R$
4      Select random token $t$ where $t \in r_i$
5      Perform random correction $c \epsilon C$ to $t$ to get $\hat{r}_j$
6      Append $\hat{r}_j$ to $\hat{R}$
7  end
8  for j = 1 to k do
9      Classify: $f(\hat{r}_j)$
10     Append results to predictions array $P$
11 end
12 $\hat{f}(R) = V(P)$, $V()$ is a voting function
13 if $f(\hat{R}) == negative$ then
14     $T = count(f(\hat{r}_{ij}) == negative) / N * l$
15 else
16     $T = count(f(\hat{r}_{ij} == positive) / N * l$
17 end
```

for Increased Randomness Defense, while Random Perturbations Defense makes k random corrections to get the final replicate $\hat{r}_{ij}$.

3.3 Overcoming the Attacks

We define an attack as making random perturbations to an input, specifically for this work, a review R. We assume a uniform distribution for randomness. We interpret these random changes to occur throughout each review R with probability $\frac{1}{W}$ or $\frac{1}{N*m}$, where W is the number of words in R, N is the number of sentences in R, and m is the average length of each sentence in R. We refer to this probability that an attack makes changes to the review text as P_{attack} where a is the total number of perturbations made by the attack:

$$P_{attack} = \frac{a}{W} = \frac{a}{N*m}.$$

If each random perturbation performed by the attack has a probability of $\frac{1}{N*m}$, then our defense method needs to overcome that probability to overcome the attack.

Our two defense methods, Random Perturbations Defense and Increased Randomness Defense, both offer ways to overcome the attack, i.e., undo the attack change, with a probability greater than $\frac{a}{N*m}$.

Proposition 1 *Random Perturbations Defense overcomes an attack that makes a small number of random perturbations to a review document by having a probability greater than the attack probability* P_{attack}.

Our Random Perturbations Defense picks a random token t from each sentence $r_i \in R$ and repeats k

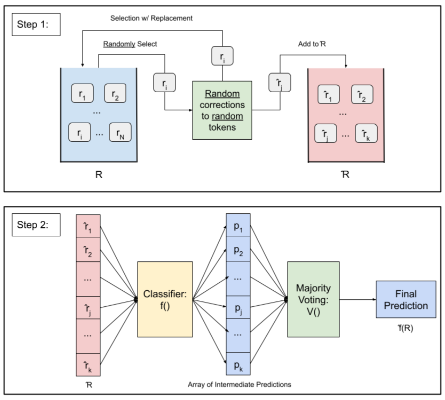

Step 1:
Selection w/ Replacement
Randomly Select
Add to $\hat{R}$
r_1 r_2 ... r_i ... r_N
r_i
Random corrections to random tokens
$\hat{r}_j$
$\hat{r}_1$ $\hat{r}_2$ ... $\hat{r}_j$ ... $\hat{r}_k$
R
$\hat{R}$
Step 2:
$\hat{r}_1$ $\hat{r}_2$ ... $\hat{r}_j$ ... $\hat{r}_k$
Classifier: f()
p_1 p_2 ... p_j ... p_k
Majority Voting: V()
Final Prediction
$\hat{f}(R)$
$\hat{R}$
Array of Intermediate Predictions

Figure 2: Visual representation of Algorithm 2 that includes more randomness.

times to get a final replicate $\hat{r}_{ij}$. This gives an initial probability that the defense picks a certain token from the text, or P_{RPD}, to be:

$$P_{RPD} = \frac{N * l * m!}{k!(m-k)!}.$$

We find this probability from choosing k tokens from r_i with length m which breaks down to a binomial coefficient $\binom{m}{k} = \frac{m!}{k!(n-k)!}$. This is then repeated l times for each sentence in R which equates to that initial probability being multiplied by l and N. After doing some rearranging of the probabilities, we can see that for certain values of l and k where $k < m$:

$$\mathrm{P}_{RPD} = \frac{N^2 m^2 l(m-1)(m-2)...(m-k+1)}{k!} > a.$$

P_{RPD} now is the total probability that the defense makes random changes to lN tokens. We know that $W = N * m$, that $a =< W$ for the attack methods we are testing against, and that k should be selected so that $k << W$. This means that we know $W^2 > a$, $W^2 > k!$, and $l(m-1)(m-2)...(m-k+1) > 0$ for the selected attack methods, which gives us the necessary conditions to assert that $P_{RPD} > Pattack$. Therefore, our Random Perturbations Defense will overcome the P_{attack} and should overcome the given attack method as stated in Proposition 1.

Proposition 2 *Increased Randomness Defense overcomes an attack that makes a small number of random perturbations to a review document by having a probability greater than the attack probability* P_{attack}.

Our Increased Randomness Defense first chooses a random sentence r_i which is selected with probability $\frac{1}{N}$. Next, we choose a random word within that sentence which is selected with probability $\frac{1}{m}$. This gives us a probability for changes as follows:

$$P_{IRD} = \frac{1}{N} * \frac{1}{m} = \frac{1}{N * m}.$$

We can see that $P_{IRD} * a = P_{attack}$. We need to overcome the attack probability and we do this in two ways: we either find the attack perturbation by chance and reverse it, or we counterbalance the attack perturbation with enough replicates $\hat{r}_j$. With each replicate $\hat{r}_j$ created, we increase our probability P_{IRD} so that our final probability for our Increased Randomness Defense is as follows:

$$P_{IRD} = \frac{k}{N * m}.$$

As long as our selected parameter value for k is greater than the number of perturbation changes made by the attack method a, then $P_{IRD} > P_{attack}$ and our Increased Randomness Defense method will overcome the given attack method as stated in Proposition 2.

4 Experiments & Results

4.1 Dataset & Models

We used the IMDB dataset (Maas et al., 2011) for our experiments. Each attack was used to perturb 100 reviews from the dataset. The 100 reviews were selected randomly from the dataset with a mix of positive and negative sentiments. Note that the Kuleshov attack data (Kuleshov et al., 2018) only had 77 reviews.

The models used in this research are from the TextAttack (Morris et al., 2020) and HuggingFace (Wolf et al., 2020) libraries. These libraries offer many different models to use for both attacked data generation and general NLP tasks. For this research, we used the *bert-base-uncased-imdb* model that resides in both the TextAttack and HuggingFace libraries. This model was fine-tuned and trained with a cross-entropy loss function. This model was used with the API functions of the TextAttack library to create the attacked reviews from each of the attacks we used. We chose this model because BERT models are useful in many NLP tasks and this model specifically was fine-tuned for text classification and was trained on the dataset we wanted to use for these experiments.

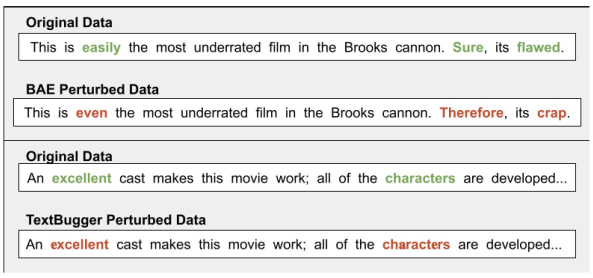
The HuggingFace library was also used in the sentiment-analysis classification of the attacked data and the defense method. We used the HuggingFace transformer pipeline for sentiment-analysis to test our defense method. This pipeline returns either “negative” or “positive” to classify the sentiment of the input text and a score for that prediction (Wolf et al., 2020). This pipeline was used to classify each replicate $\hat{r}_{ij}$ in our algorithm and is represented as the function $f()$.

4.2 Experiments

The attacks from the TextAttack library were used to generate attack data. Attack data was created from 7 different models from the library: BERT-based Adversarial Examples (BAE) (Garg and Ramakrishnan, 2020), DeepWordBug (Gao et al., 2018), FasterGeneticAlgorithm (Jia et al., 2019), Kuleshov (Kuleshov et al., 2018), Probability Weighted Word Saliency (PWWS) (Ren et al., 2019), TextBugger (Li et al., 2019), and TextFooler (Jin et al., 2020) (Morris et al., 2020). Each of these attacks were used to create 100 perturbed sentences from the IMDB dataset (Maas et al., 2011). These attacks were chosen from the 14 classification model attacks because they represent different kinds of attack methods, including misspelling, synonym substitution, and antonym substitution.

Each attack method used for our experiments has a slightly different approach to perturbing the input data. Each perturbation method is unique and follows a specific distinct pattern and examples of these can be found in Figure 3. The BAE attack determines the most important token in the input and replaces that token with the most similar replacement using a Universal Sentence Encoder. This helps the perturbed data remain semantically similar to the original input (Garg and Ramakrishnan, 2020). The DeepWordBug attack identifies the most important tokens in the input and performs character-level perturbations on the highest-ranked tokens while minimizing edit distance to create a change in the original classification (Gao et al., 2018). The FasterGeneticAlgorithm perturbs every token in a given input while maintaining the original sentiment. It chooses each perturbation carefully to create the most effective adversarial example (Jia et al., 2019). The Kuleshov attack is a synonym substitution attack that replaces 10% - 30% of the tokens in the input with synonyms that do not change the meaning of the input (Kuleshov et al., 2018).

The PWWS attack determines the word saliency score of each token and performs synonym substitutions based on the word saliency score and the maximum effectiveness of each substitution (Ren et al., 2019). The TextBugger attack determines the important sentences from the input first. It then determines the important words in those sentences and generates 5 possible “bugs” through different perturbation methods: insert, swap, delete, sub-c (visual similarity substitution), sub-w (semantic similarity substitution). The attack will implement whichever of these 5 generated bugs is the most effective in changing the original prediction (Li et al., 2019). Finally, the TextFooler attack determines the most important tokens in the input using synonym extraction, part-of-speech checking, and semantic similarity checking. If there are multiple canididates to substitute with, the most semantically similar substitution will be chosen and will replace the original token in the input (Jin et al., 2020).



Original Data

This is easily the most underrated film in the Brooks cannon. Sure, its flawed.

BAE Perturbed Data

This is even the most underrated film in the Brooks cannon. Therefore, its crap.

Original Data

An excellent cast makes this movie work; all of the characters are developed...

TextBugger Perturbed Data

An excellent cast makes this movie work; all of the characters are developed...

Figure 3: Example of what original data looks like and how the BAE (Garg and Ramakrishnan, 2020) and TextBugger (Li et al., 2019) attack methods perturb data. The BAE attack method uses semantic similarity, while the Textbugger attack method uses visual similarity.

After each attack had corresponding attack data, the TextAttack functions gave the results for the success of the attack. The accuracy of the sentiment-analysis task under attack, without the defense method, is reported in the first column in Table 1. Each attack caused a large decrease in the accuracy of the model. The model began with an average accuracy of 80% for the IMDB dataset. Once the attack data was created and the accuracy under attack was reported, the attack data was run through our Random Perturbations and Increased Randomness defense methods. All of the experiments were run on Google Colaboratory using TPUs and the Natural Language Toolkit (Loper and Bird, 2002).

4.3 Results

We began by testing on the HuggingFace sentiment analysis pipeline with the original IMDB dataset.

This gave an original accuracy of 80%. This percentage represents the goal for our defense method accuracy as we aim to return the model to its original accuracy, or higher. The accuracy under each attack is listed in Table 1 in the first column. These percentages show how effective each attack is at causing misclassification for the sentiment analysis task. The attacks range in effectiveness with PWWS (Ren et al., 2019) and Kuleshov (Kuleshov et al., 2018) with the most successful attacks at 0% accuracy under attack and FasterGeneticAlgorithm (Jia et al., 2019) with the least successful attack at 44% accuracy under attack, which is still almost a 40% drop in accuracy.

Attack	w/o Defense	w/ Defense
BAE	33%	80.80%±1.47
DeepWordBug	34%	76.60%±1.85
FasterGeneticAlgo	44%	82.20%±1.72
Kuleshov*	0%	60.00%±2.24
PWWS	0%	81.80%±1.17
TextBugger	6%	79.20%±2.32
TextFooler	1%	83.20%±2.48

Table 1: Accuracy for each of the attack methods under attack, and under attack with the defense method from Algorithm 1 deployed with $l = 7$ and $k = 5$. The accuracy prior to attack is 80%.

4.3.1 Random Perturbations Defense

For the Random Perturbations Defense to be successful, it is necessary to obtain values of the two parameters, l and k. Each attack was tested against our Random Perturbations Defense 5 times. The accuracy was averaged for all 5 tests and the standard deviation was calculated for the given mean. The mean accuracy with standard deviation is presented for each attack in the second column of Table 1. The results presented are for $l = 7$ and $k = 5$. These parameters were chosen after testing found greater values of l and k resulted in a longer run time and too many changes made to the original input; with lower values for l and k, the model had lower accuracy and not enough perturbations to outweigh any potential adversarial attacks. The values behind this logic can be seen in Table 2.

The defense method was able to return the model to original accuracy within statistical significance while under attack for most of the attacks with the exception of the Kuleshov method (Kuleshov et al., 2018). The accuracy for the other attacks all were returned to the original accuracy ranging from 76.00% to 83.20% accuracy with the Random Perturbations defense deployed. This shows that our defense method is successful at mitigating most potential adversarial attacks on sentiment classification models. Our defense method was able to increase the accuracy of model while under attack for the FasterGeneticAlgorithm, PWWS, and TextFooler. These three attack methods with our defense achieved accuracy that was higher than the original accuracy with statistical significance.

Attack	l	k	Accuracy w/ Defense
BAE	5	2	55%
BAE	10	5	50%
BAE	7	5	79%

Table 2: This table explains values of l and k

4.3.2 Increased Randomness Defense

The Increased Randomness Defense was also tested on all seven of the attacks. Each attack was tested against this defense 5 times. The results for these experiments can be seen in Table 4. There were tests done to determine what the proper value for k should be. These tests were performed on the BAE (Garg and Ramakrishnan, 2020) attack and the results can be found in Table 3. These tests revealed that 40-45 replicates $\hat{r}_j$ was ideal for each $\hat{R}$ with $k = 41$ being the final value used for the tests on each attack. This defense method was more efficient to use.

Attack	k	Accuracy w/ Defense
BAE	10	67%
BAE	20	76%
BAE	25	72%
BAE	30	76%
BAE	35	74%
BAE	40	82%
BAE	45	74%
BAE	41	77%

Table 3: This table shows the results for the tests for different values of k for the increased randomness experiments.

The runtime and the resources used for this method were lower than the original random perturbations defense method with the runtime for the Random Perturbations Defense being nearly 4 times longer than this increased random method. A comparison of the two defense methods on the seven attacks tested can be seen in Figure 4. This defense was successful in returning the model to the original accuracy, within statistical significance, for most of the attacks with the exception of the Kuleshov attack (Kuleshov et al., 2018). A t-test was performed to determine the statistical significance of the difference in the defense method accuracy to the original accuracy.

Attack	w/o Defense	w/ Defense
BAE	33%	78.40%±3.14
DeepWordBug	34%	76.80%±2.64
FasterGeneticAlgo	44%	82.80%±2.48
Kuleshov*	0%	66.23%±4.65
PWWS	0%	79.20%±1.72
TextBugger	6%	77.00%±2.97
TextFooler	1%	80.20%±2.48

Table 4: Accuracy for increased randomness defense from Algorithm 2 against each attack method with $k = 41$. The accuracy prior to attack is 80%.

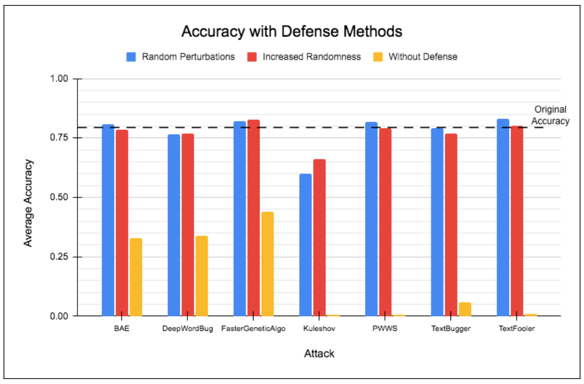


Figure 4: Comparing the average accuracy of the Random Perturbations Defense and the Increased Randomness Defense methods to the under attack accuracy without defense on the seven attacks.

4.4 Comparison to Recent Defense Methods

Our defense methods are comparable to some recent defense methods created for text classification. Our defense method returns the model to the original accuracy within statistical significance. This is comparable to the work done by Zhou et al. (2021) in their Dirichlet Neighborhood Ensemble (DNE) defense method. They were able to bring the model within 10% of the original accuracy for CNN, LSTM, and BOW models for the IMDB dataset. However, their work is only applicable to synonym-substitution based attacks. Since our defense methods apply equally well to seven attacks, it is general and can be applied without determining the exact type of attack (assuming it is one of the seven).

Another recent defense method, Synonym Encoding Method (SEM), was tested on synonym-substitution attacks on Word-CNN, LSTM, Bi-LSTM and BERT models (Wang et al., 2021b). This defense method was most successful on the BERT model and was able to return to the original accuracy within 3% for the IMDB dataset. Our work is comparable to both DNE and SEM which represent recent work in defending NLP models against adversarial attacks and more specifically synonym-substitution based attacks.

WordDP is another recent defense method for adversarial attacks against NLP models (Wang et al., 2021a). This defense method used Differential Privacy (DP) to create certified robust text classification models against word substitution adversarial attacks. They tested their defense on the IMDB and found that their WordDP method was successful at raising the accuracy within 3% of the original clean model. This method outperformed other defense methods including DNE. This is similar to our defense method, but they do not include whether these results are statistically significant.

We also compare our defense methods, RPD and IRD, against these recent defense methods on cost and efficiency. Our RPD and IRD methods have comparable time complexity of $O(cn)$, where c is the time it takes for classification and n is the number of reviews. Each method has a similar constant that represents the number of perturbations and replicates made. We cannot directly compare the time complexity of our defense methods with the SEM, DNE, and WordDP methods. These recent defense methods require specialized training and/or encodings. Our RPD and IRD methods do not require specialized training or encodings, so they cannot be directly compared on time complexity. This means that the comparison between our methods and recent defense methods comes in the form of specialized training vs. input preprocessing. Training and developing new encodings tends to be more time consuming and expensive than input preprocessing methods that can occur during the testing phases.

5 Conclusion

The work in this paper details a successful defense method against adversarial attacks generated from the TextAttack library. These attack methods use multiple different perturbation approaches to change the predictions made by NLP models. Our Random Perturbations Defense was successful in mitigating 6 different attack methods. This defense method returned the attacked models to their original accuracy within statistical significance. Our second method, Increased Randomness Defense, used more randomization to create an equally successful defense method that was 4 times more efficient than our Random Perturbations Defense. Overall, our defense methods are effective in mitigating a range of NLP adversarial attacks, presenting evidence for the effectiveness of randomness in NLP defense methods. The work done here opens up further study into the use of randomness in defense of adversarial attacks for NLP models including the use of these defense methods for multi-class classification. This work also encourages a further mathematical and theoretical explanation to the benefits of randomness in defense of NLP models.

Acknowledgement

The work reported in this paper is supported by the National Science Foundation under Grant No. 2050919. Any opinions, findings and conclusions or recommendations expressed in this work are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

2019. Predicting the direction of stock market prices using tree-based classifiers. *The North American Journal of Economics and Finance*, 47:552–567.

Basemah Alshemali and Jugal Kalita. 2020. Generalization to mitigate synonym substitution attacks. In *Proceedings of Deep Learning Inside Out (DeeLIO): The First Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 20–28.

Michel Ballings, Dirk Van den Poel, Nathalie Hespeels, and Ruben Gryp. 2015. Evaluating multiple classifiers for stock price direction prediction. *Expert systems with Applications*, 42(20):7046–7056.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32.

Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. 2019. Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4658–4664.

Xiaoyi Chen, Ahmed Salem, Michael Backes, Shiqing Ma, and Yang Zhang. 2021. Badnl: Backdoor attacks against nlp models. In *ICML 2021 Workshop on Adversarial Machine Learning*.

John P Corradi, Stephen Thompson, Jeffrey F Mather, Christine M Waszynski, and Robert S Dicks. 2018. Prediction of incident delirium using a random forest classifier. *Journal of Medical Systems*, 42(12):1–10.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *2019 Annual Conference of the National American Chapter of the Association for Computational Linguistics: Human Language Technologies*, page 4171–4186.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication. MIT Press, Cambridge, MA.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56.

Siddhant Garg and Goutham Ramakrishnan. 2020. BAE: BERT-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181.

Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Madry, Bo Li, and Tom Goldstein. 2020. Data security for machine learning: Data poisoning, backdoor attacks, and defenses. *arXiv preprint arXiv:2012.10544*.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. *stat*, 1050:20.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 2019. Certified robustness to adversarial word substitutions. 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 4129–4142.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

Mohammed Khalilia, Sounak Chakraborty, and Mihail Popescu. 2011. Predicting disease risks from highly imbalanced data using random forest. *BMC Medical Informatics and Decision Making*, 11(1):1–13.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar. Association for Computational Linguistics.

Volodymyr Kuleshov, Shantanu Thakoor, Tingfung Lau, and Stefano Ermon. 2018. Adversarial examples for natural language classification problems. In *Open Review Net*.

AV Lebedev, Eric Westman, GJP Van Westen, MG Kramberger, Arvid Lundervold, Dag Aarsland, H Soininen, I Kłoszewska, P Mecocci, M Tsolaki, et al. 2014. Random forest ensembles for detection and prediction of alzheimer's disease with a good between-cohort robustness. *NeuroImage: Clinical*, 6:115–125.

Alexander Levine and Soheil Feizi. 2020. Robustness certificates for sparse adversarial attacks by randomized ablation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4585–4593.

J Li, S Ji, T Du, B Li, and T Wang. 2019. Textbugger: Generating adversarial text against real-world applications. In *26th Annual Network and Distributed System Security Symposium*.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia*, pages 69–72.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, Oregon, USA.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126.

Kofi O Nti, Adebayo Adekoya, and Benjamin Weyori. 2019. Random forest based feature selection of macroeconomic variables for stock market prediction. *American Journal of Applied Sciences*, 16(7):200–212.

Desbordes Paul, Ruan Su, Modzelewski Romain, Vauclin Sébastien, Vera Pierre, and Gardin Isabelle. 2017. Feature selection for outcome prediction in oesophageal cancer using genetic algorithm and random forest classifier. *Computerized Medical Imaging and Graphics*, 60:42–49.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014*.

Eric Wallace, Tony Zhao, Shi Feng, and Sameer Singh. 2021. Concealed data poisoning attacks on NLP models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 139–150.

Wenjie Wang, Pengfei Tang, Jian Lou, and Li Xiong. 2021a. Certified robustness to word substitution attack with differential privacy. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Xiaosen Wang, Hao Jin, Yichen Yang, and Kun He. 2021b. Natural language adversarial defense through synonym encoding. In *Conference on Uncertainty in Artificial Intelligence*.

Zhaoyang Wang and Hongtao Wang. 2020. Defense of word-level adversarial attacks via random substitution encoding. In *Knowledge Science, Engineering and Management*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in Neural Information Processing Systems*, 28:649–657.

Yi Zhou, Xiaoqing Zheng, Cho-Jui Hsieh, Kai-Wei Chang, and Xuanjing Huang. 2021. Defense against synonym substitution-based adversarial attacks via Dirichlet neighborhood ensemble. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Liuwan Zhu, Rui Ning, Cong Wang, Chunsheng Xin, and Hongyi Wu. 2020. Gangsweep: Sweep out neural backdoors by gan.

Retrofitting of Pre-trained Emotion Words with VAD-dimensions and the Plutchik Emotions

Manasi Kulkarni[1,2]
[2]Computer Engineering and IT
Veermata Jijabai Technological Institute
Mumbai
manasi@cse.iitb.ac.in

Pushpak Bhattacharyya[1]
[1]Computer Science and Engineering
Indian Institute of Technology Bombay
Mumbai
pb@cse.iitb.ac.in

Abstract

The word representations are based on distributional hypothesis according to which words that occur in the similar contexts, tend to have a similar meaning and appear closer in the vector space. For example, the emotionally dissimilar words ”joy” and ”sadness” have higher cosine similarity. The existing pre-trained embedding models lack in emotional words interpretations. For creating our VAD-Emotion embeddings, we modify the pre-trained word embeddings with emotion information. This is a lexicons based approach that uses the Valence, Arousal and Dominance (VAD) values, and the Plutchik's emotions to incorporate the emotion information in pre-trained word embeddings using post-training processing. This brings emotionally similar words nearer and emotionally dissimilar words away from each other in the proposed vector space. We demonstrate the performance of proposed embedding using NLP downstream task - Emotion Recognition.

1 Introduction

An emotion is a feeling that characterizes the state of mind such as happiness, sadness, anger, fear and more. The emotions are classified using various taxonomies under the dimensional models and the psychological emotion models such as Ekman (1992) Emotion Model , Plutchik (1980) Emotion Wheel, Parrot (2001) Model which agree to a basic set of emotion with few changes. The Plutchik emotion model represents the categorization of emotion words into 8 basic emotions : anger, anticipation, disgust, fear, joy, sadness, surprise, and trust. The PAD emotional state model (Mehrabian, 1994) is a 3-dimensional model that represents every emotion in Valence (Pleasure), Arousal and Dominance dimensions.

Emotion detection in the text is critical for a number of applications and services in diverse domains, including market research, customer-care, psychological healthcare, and intelligent tutoring systems and so on.(Mohammad and Turney, 2013). The automatic detection of emotions remains a challenging task till date as researchers may use different emotion models with different number and types of emotion categories. Also, the emotions are subjective, hence creation of emotion related resources requires much time and effort.

Word embedding are distributed word representations where each word w in the vocabulary V is mapped into a dense, low-dimensional, continuous valued vector $v_w \epsilon R^d$. Here d represents dimensions of the vector space model. Most of the embeddings are modeled using the syntactic context of words which means words appearing in the similar contexts have the similar semantics and appear closer in the vector space (Mikolov et al., 2013), (Pennington et al., 2014). As a consequence, emotionally opposite words, such as “joy” and “sorrow” occurring in similar contexts show higher cosine similarity. Hence, said property does not fit in case of emotion words as 'joy' and 'sorrow' and many more similar and opposite emotion words too.

We propose a model that modifies the pre-trained word embedding with emotion information using a post-training processing method. This is a lexicons based approach that uses the Valence, Arousal and Dominance (VAD) values, and the Plutchik's emotions to incorporate the emotion information in the word embeddings.

The main contributions of this paper includes the creation of emotion-fitted embedding to be used for NLP downstream tasks related to emotion analysis. We present average of cosine similarities for emotion words. The visualizations shown for NRC-emolex lexicons using for Glove-300d embeddings, and retrofitted embeddings at both steps : VAD-append embeddings and VAD-Emotion embeddings confirms the step-by-step clustering of similar emotion words. The accuracy for emotion recognition as downstream emotion task using the proposed embedding is mentioned as results. Though clustering of emotion words takes place, the accuracy for emotion recognition task using proposed embedding is closer to baseline but can not outperform the same.

The paper is organized as follows. Section 2 discusses various approaches used by researchers for updating vector space models for specific set of NLP tasks. Section 3 describes the proposed approach. experiment and setting are explained in section 4. We discuss the results and observation in section 5.

2 Related Work

The use of lexical semantic information (lexical resources), sentiment information, emotion information to improve distributional representations in respective area of NLP tasks is recent. Methods like Tang et al. (2016), Agrawal et al. (2018), Ye et al. (2018) achieve improved representations by using training on unlabelled corpora, distant supervision and other techniques to gain relational knowledge to modify the prior or add a regularization term. Such methods are known as 'pre-training methods', as they alter the training process for word representations. Such methods may require a change in the loss function with training and may be computationally expensive.

Ye et al. (2018) proposed a method for sentiment analysis, where they use external knowledge from SentiWordNet (Baccianella et al., 2010), Extended ANEW lexicons (Warriner et al., 2013) with pre-trained word embeddings during joint parameter training to a CNN classifier using training data. Agrawal et al. (2018) proposed a distant supervision method for automatically labeling a large corpus of training data with fine-grained emotions; and the LSTM model architectures for learning emotion-enriched word embedding from this training data.

On the other hand post-training methods include external information to modify to the vanilla word representations such as Word2Vec, GLOVE to name a few. Retrofitting Method (Faruqui et al., 2015), has used word relation knowledge from semantic lexicons (e.g. WordNet), to bring similar words closer in the retrofitted vector space. It injects antonym and synonym constraints to improve the existing word representations. Mrkšić et al. (2016) presented post-training approach named as counter-fitting which injects antonym and synonym constraints into existing vector space representations in order to improve the vectors' capability for increase semantic similarity.

Aff2Vec (Khosla et al., 2018) aims at incorporating affective information in word representations. They have used the Warriner's VAD lexicon (Warriner et al., 2013) to improve the strength in the antonym-synonym relationships of the words is incorporated to the word distribution space. The word similarity task and other sentiment analysis related tasks are performed to display the results.

Seyeditabari et al. (2019) proposed a method, based on counterfitting approach (Mrkšić et al., 2016) that incorporates emotional information of words into the model. It uses an NRC-emotional lexicon (Mohammad and Turney, 2013) and the Plutchik's model of basic emotions to prepare emotion constraints for fitting an emotional information into pre-trained word vectors.

Here, we propose a pipeline model to integrate Valence, Arousal and Dominance information of emotion words and their basic emotion labels from lexicon set to create a new vector space on pre-trained word vectors using post-training (post-processing) method.

3 Fitting VAD values and Emotion constraints into the Word Embedding

This work aims at incorporating affect information in word representations. The figure - 1 illustrates overview of proposed system for retrofitting of pre-trained vector space. The emotion information is infused in two consecutive steps. The subsections 3.1 and 3.2 discuss the processing performed in step-1 and step-2 respectively. As shown in figure-1, at step-1, we append the Valence, Arousal and Dominance (VAD) values for the emotion lexicons to their respective word embedding to transform an original pre-trained vector space V into the VAD-appended vector space V'. In step-2, we use emotion constraints to modify these VAD-appended embeddings further to achieve retrofitted vector space V". Following subsections present the steps of modifications in the pre-trained word vector space V to V' and then to the final modified vector space V" from V'.

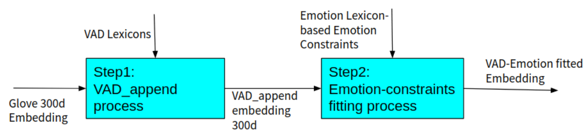

VAD Lexicons
Emotion Lexicon-based Emotion Constraints
Glove 300d Embedding
Step1: VAD_append process
VAD_append embedding 300d
Step2: Emotion-constraints fitting process
VAD-Emotion fitted Embedding

Figure 1: The System Overview

NRC-VAD lexicons : NRC-VAD lexicon (Mohammad, 2018) is a set of affect lexicons with 20007 English words. It contains real-valued scores for valence, arousal, and dominance (VAD) on a scale of [0 - 1] for each lexicon.

3.1 Step-1 : VAD-Append

Consider the word embedding space V and the affect ([V,A,D]) embedding space A. The word vector v_w of word w , $v_w \epsilon R^M$, is concatenated with the respective VAD-vector $a_w = [V, A, D] \epsilon R^3$ from A, resulting in a $M + 3$ dimensional word representation (Khosla et al., 2018). For the words not present in VAD-NRC lexicons, $a_w = [0.5, 0.5, 0.5] \epsilon R^3$ is assumed. Then these vectors are reduced to M dimensions using dimensionality reduction algorithm such as Principle Component Analysis (PCA) so that their performance can be compared with existing pre-trained embedding. This process transforms a pre-trained vector space V to VAD-appended vector space V' as shown in figure-2.

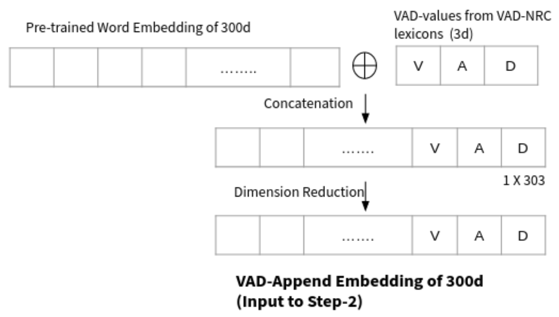

Pre-trained Word Embedding of 300d
....
VAD-values from VAD-NRC lexicons (3d)
V
A
D
Concatenation
....
1 X 303
Dimension Reduction
VAD-Append Embedding of 300d (Input to Step-2)

Figure 2: The VAD-Append Process for a word

3.2 Step-2 : Emotion constraints using the Plutchik Model of emotions

For fitting emotional information into VAD-Append word vectors, we use a methodology on

Emotion-1	**Emotion-2**	**Differing dimension from VAD**
Anger	Fear	D
Anticipation	Surprise	AD
Disgust	Trust	VD
Joy	Sadness	VAD

Table 1: VAD-dimensions and opposite emotions

the similar lines of (Seyeditabari et al., 2019). We aim to modify VAD-Append vector space $V' = \{v_1', v_2', \ldots, v_n'\}$ to new vector space $V" = \{v_1", v_2", \ldots, v_n"\}$ to add emotion information without loosing much information present with original vectors.

The Plutchik's emotion model defines 4 pairs of opposite emotions: Anger and Fear, Disgust and Trust, Anticipation and Surprise, and Joy and Sadness. These emotions differ based on high(1) or low(0) VAD-values for the respective emotions. Anger and Fear differs on Dominance value as Dominance is high for Anger and low for Fear. Table-1 shows difference for rest of the emotions.

We refine the VAD-append vectors further to increase the cosine similarity between words with similar emotions and decrease cosine similarity between the dissimilar emotion words which can help improve new vector space having word vectors with interpretation of emotions in the respective words.

To achieve this, two emotion constraint lists are created. First list $TrueEmotion$ maintains pairs as (word, true_emotion) for every lexicon from NRC-Emolex such as $\{(w_1, e_1), (w_1, e_2) \ldots (w_2, e_1), \ldots (w_n, e_3) \ldots\}$ and another list $OppositeEmotion$ maintains pairs as (word, opposite_emotion) for every pair present in the first list such as $\{(w_1, o_1), (w_1, o_2) \ldots (w_2, o_1) \ldots (w_n, o_3) \ldots \}$. Here o_i represents the opposite emotion of i^{th} emotion e_i as shown on the Plutchik wheel of emotions. The NRC-Emolex lexicons (Mohammad and Turney, 2013) are annotated with the best suitable Plutchik emotions and positive or negative as sentiment value are used for the same.

Our objective function for step-2 is based on the counterfitting approach (Mrkšić et al 2016) to decrease the cosine distance between words with their associated emotion in the list

$TrueEmotion(TE)$, and to increase cosine distance with their opposite emotions in the list $OppositeEmotion(OE)$. The objective function is to be minimised to achieve proposed vector space model V'. The objective function contains following three terms,

$$\begin{aligned} Obj(V^{'}, V^{"}) = c_1 OR(V^{"}) + c_2 TA(V^{"}) \\ + c_3 VSP(V^{'}, V^{"}) \end{aligned} \tag{1}$$

Opposite Repels(OR) : This term is to reduce cosine similarity between words' and opposite emotions' vectors away from each other in by increasing cosine distance between them in the transformed vector space V'. It uses the $OppositeEmotion$ list for this purpose.

$$OR(V^{"}) = \sum_{(w,o)\epsilon OE} max(0, \delta - d(v_w^{"}, v_o^{"})) \tag{2}$$

Here, standard value of $\delta = 1$ and cosine distance $d(v_w, v_o) = 1 - cos_{dist}(v_w, v_o)$

True Attracts(TA): This term is to bring the embeddings of the words and their respective true emotions nearer to each other in new vector space V'. In other words, for increasing cosine similarity between them.

$$TA(V^{"}) = \sum_{(w,e)\epsilon TE} max(0, d(v_w^{"}, v_e^{"}) - \gamma)) \tag{3}$$

The $\gamma = 0$ represents minimum distance between true emotion and words.

Vector Space Preservation(VSP) : The original vector space describes the distributional information for words from very large textual corpora (Mrkšić et al., 2016). VSP term tries to minimize the difference between cosine distance between word pairs in original vector space V and new vector space V' to preserve semantic and contextual information as much as possible. In current experiments, the neighbouring words are chosen from NRC-Emolex only.

$$\begin{aligned} VSP(V^{'}, V^{"}) = \\ \sum_{i=1}^{N} \sum_{j\epsilon N(i)} (max(0, d(v_x^{"}, v_y^{"}) - d(v_x^{'}, v_y^{'}))) \end{aligned} \tag{4}$$

.

4 Experiments

We have used the NRC-VAD lexicons (Mohammad, 2018) to create VAD vectors and the NRC-Emolex emotion lexicons (Mohammad and Turney, 2013) for emotion constraints creation as they are labelled with Plutchik's wheel of emotions (Plutchik, 1980) along with positive and negative sentiment values.

During experiments, Glove-300d (Pennington et al., 2014), FastText-1M (Joulin et al., 2016), and Google Word2Vec (Mikolov et al., 2013) are used as pre-trained input vector space and created the VAD-append embeddings at step-1, VAD-Emotion word embeddings at step-2 respectively. We run Stochastic Gradient Descent (SGD) for 20 epochs to achieve the final retrofitted embeddings with emotion information.

We have compared performance of VAD-Emotion fitted embeddings with existing vector space as mentioned above for the task of finding average cosine similarity over NRC-Emotion lexicons for similar and opposite emotions. Also, have displayed accuracy results of Emotion Recognition task on ISEAR dataset (ISEAR) and the Twitter Emotion Corpus (TEC) dataset (Mohammad, 2012).

5 Results and Discussion

Table-2 shows average cosine similarity between NRC-Emolex lexicons and their respective emotions. Higher the cosine similarity is better for the similar emotion words. For Joy and Trust emotions, VAD-Emotion Embedding perform better than rest of them.

Table-3 shows average cosine similarity between NRC-Emolex lexicons and the opposite emotions.The cosine similarity between lexicons and emotion should be as low as possible. The range for cosine similarity is $[+1, -1]$ The figure - 5 shows the clustering of similar emotion words together and opposite emotion words in opposite word clusters.

Table-4 shows accuracy of Emotion Recognition Task using the pre-trained word embedding, and respective VAD-append and VAD-Emotion Embedding as input to a simple BiLSTM model. The Table-4 reports 4-fold cross validation accuracy on the subset of ISEAR dataset with

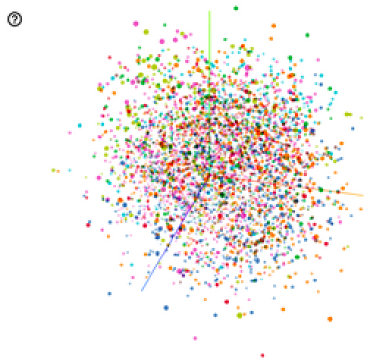


Figure 3: Visualization of NRC-Emolex lexicons using Glove-300d embeddings

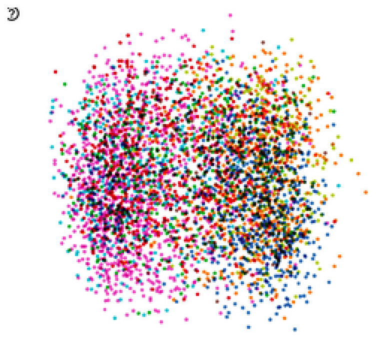


Figure 4: Visualization of NRC-Emolex lexicons using VAD-Append embeddings after step-1 processing

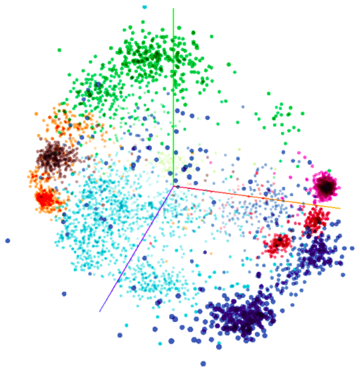


Figure 5: Clustering of NRC-Emolex lexicons using VAD-dimention + Plutchik emotion embeddings after step-2 processing

5.4k examples and TEC dataset with 20k examples respectively used for training and testing.

For the purpose of comparison, the emotion recognition task on the ISEAR dataset was performed with the Emotion-Refined-Embedding (Seyeditabari et al., 2019) with Glove-300d pre-trained embedding which resulted as accuracy of **64.84%**. The embeddings are computed with help of the code of Seyeditabari et al. (2019) which is available publicly.

It can be observed from emotion recognition accuracy values that VAD-Append embedding give better accuracy than pre-trained Embedding - Glove, Google Word2Vec, FastText as well as respective VAD-Emotion embedding. The initial segregation based on V,A,D values helps to achieve improvement in accuracy of emotion recognition.

The figure-3, figure-4, and figure-5 show visualization for NRC-Emolex lexicons using Glove-300d embeddings, VAD-append embeddings (step-1 output) and VAD-Emotion embeddings i.e. final proposed retrofitted embeddings. It can be observed that VAD-Emotion embeddings show the better cosine similarity among the emotionally similar words and less cosine similarity between emotionally dissimilar words than rest of them. This can be confirmed by looking at the clusters of emotion lexicons formed in figure - 5. Yet, the accuracy for emotion recognition, do not show better results for VAD-Emotion Embedding.

The reason from the primary observation is that it is due to overlaps in clusters, as one lexicon may belong to one or more emotions. Also, at step-2, we retrofit word embeddings, only for emotion words present in NRC-Emolex. The limitation with post-processing methods such as counter-fitting is that it retrofits emotion words present in the constraints. Hence, we should perform a global specialization or post-specialization processing (Vulic et al., 2018) for retrofitting non-emotion words with reference to the newly retrofitted emotion word vectors. This will turn into the retrofitting of complete vector space with the Plutchik's emotion information, which may help to improve accuracy of the downstream tasks.

At step-1 every word embedding is appended with VAD-values or the neutral value vector [0.5,0.5,0.5] which retrofits every word for VAD-values in V' vector space. Due to the limitation mentioned above, step-2 does not perform better than step-1. Hence, the words which are not in the NRC-lexicons can not be retrofitted at step-2. To achieve the better accuracy results with final retrofitted vector space V'', this is going to be our future work with the proposed method.

Conclusion and Future Work

Embedding models have an important role in word representation in various natural language processing tasks. Here we present an approach to bring emotionally similar words nearer and emotionally dissimilar words away from each other in the proposed vector space. This can be observed in the better cosine similarity results presented in table - 2, 3 for the NRC-Emolex lexicons and also in the in figure - 5 that shows the similar emotion words from Emolex are clustered together and opposite emotion words are far apart. In this

Lexicons labelled with Emotion	**GloVe-300d Embedding**	**VAD-append Glove Embedding (Step-1 of Proposed Approach)**	**VAD-Emotion Embedding (Step-2 of Proposed Approach)**
Anger	0.2864	0.4650	0.2548
Anticipation	0.3746	0.5496	**0.5743**
Disgust	0.2789	0.4436	**0.7646**
Fear	0.3057	0.4945	0.4023
Joy	0.3315	0.4585	**0.7720**
Sadness	0.2603	0.4329	**0.7214**
Surprise	0.2818	0.4287	0.3906
Trust	0.2567	0.4246	**0.8724**

Table 2: Average on Cosine Similarity between lexicons and their emotion labels from NRC-Emolex

Lexicons labelled with Emotion	**Opposite Emotion**	**GloVe-300d Embedding**	**VAD-append Glove Embedding (Step-1 of Proposed Approach**	**VAD-Emotion Embedding (Step-2 of Proposed Approach)**
Anger	Fear	0.3091	0.4329	**0.0181**
Anticipation	Surprise	0.2519	0.3528	**-0.0235**
Disgust	Trust	0.1625	0.1843	**-0.2284**
Fear	Anger	0.2557	0.5078	**-0.0127**
Joy	Sadness	0.2372	0.3118	**-0.1035**
Sadness	Joy	0.2029	0.2719	**-0.1164**
Surprise	Anticipation	0.2422	0.4145	**-0.0223**
Trust	Disgust	0.1626	0.2656	**-0.2264**

Table 3: Average on Cosine Similarity between lexicons from NRC-Emolex and their opposite emotions

Word Embedding	**ISEAR dataset**			**Twitter Emotion Corpus (TEC)**		
	Pre-Trained Embedding	**VAD-Append Embedding (step-1)**	**VAD-Emotion Embedding (step-2)**	**Pre-Trained Embedding**	**VAD-Append Embedding (step-1)**	**VAD-Emotion Embedding (step-2)**
Glove-300d	70.57	***70.95***	66.11	56.44	***57.62***	54.73
Google Word2Vec	69.7	***70.50***	65.84	56.90	***58.51***	53.81
FastText-1M	67.79	***70.60***	66.11	55.35	***57.70***	55.11

Table 4: Average accuracy of Emotion Recognition model for 4-fold cross validation on ISEAR dataset and TEC dataset

approach, we have combined effect of dimensional emotion model through VAD values as well as effect of psychological emotion model through emotion constraints for incorporating emotion information in the proposed vector space VAD-Emotion embedding.

However, the VAD-append embedding shows better results than pre-trained embedding, the accuracy of VAD-Emotion embedding for emotion recognition task is not impressive. The post-specialization or global-specialization process for retrofitting of non-emotion word embeddings may improve the accuracy at step-2 of proposed model.

As a future work, we will be performing a process for retrofitting non-emotion words with reference to the newly retrofitted word vectors, which may help to improve accuracy of the downstream emotion-based NLP tasks. Also, use of The knowledge base such as ConceptNet, SenticNet etc may help in finding more emotion words and emotion constraints to gain better emotion information. Being the lexicon-based approach, the approach with modifications may be useful for emotion/sentiment based applications with low-resource languages too.

References

Ameeta Agrawal, Aijun An, and Manos Papagelis. 2018. Learning emotion-enriched word representations. In *Proceedings of the 27th International Conference on Computational Linguistics, Santa Fe, New Mexico, USA*, pages 950–961.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2010).*

Paul Ekman. 1992. An argument for basic emotions. *Cognition Emotion, 6(3-4)*, pages 169–200.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615.

ISEAR. The international survey on emotion antecedents and reactions as a python dataset for machinelearning http://www.affective-sciences.org/researchmaterial.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Paper*, pages 427–431.

Sopan Khosla, Niyati Chhaya, and Kushal Chawla. 2018. Aff2vec: Affect–enriched distributional word representations. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2204–2218.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems.*

Saif Mohammad. 2012. emotional tweets. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics (*Sem), Montreal, Canada.*

Saif M Mohammad. 2018. Obtaining reliable human ratings of valence, arousal, and dominance for 20,000 english words. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 174–184.

Saif M Mohammad and Peter D Turney. 2013. Crowd-sourcing a word–emotion association lexicon. In *Computational Intelligence, 29(3)*, pages 436–465.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of NAACL-HLT 2016*, pages 142–148.

W Parrot. 2001. Emotions in social psychology. key readings in social psychology. In *Philadelphia: Psychology Press. ISBN 978-0863776830.*

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)*, page 1532–1543.

Robert Plutchik. 1980. Emotion: Theory, research, and experience. In *Vol. 1. Theories of emotion, 1, New York: Academic.*

Armin Seyeditabari, Narges Tabari, Shefie Gholizade, and Wlodek Zadrozny. 2019. Emotional embeddings: Refining word embeddings to capture emotional content of words. In *arXiv:1906.0011v2, June 2019.*

D. Tang, F. Wei, B. Qin, N. Yang, T. Liu, and M. Zhou. 2016. Sentiment embeddings with applications to sentiment analysis. In *IEEE Transactions on Knowledge and Data Engineering 28, 2(2016)*, pages 496–509.

Ivan Vulic, Goran Glavaš, Nikola Mrkšic, and Anna Korhonen. 2018. Post-specialisation: Retrofitting vectors of words unseen in lexical resources. In *Proceedings of NAACL-HLT 2018, New Orleans, Louisiana, June 1 - 6, 2018*, pages 516–527.

Xiangyu Wang and Chengqing Zong. 2021. Distributed representations of emotion categories in emotion space. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2364–2375.

Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. 2013. Norms of valence, arousal, and dominance for 13,915 english lemmas. In *Behavior Research Methods, 45(4)*, pages 1191–1207.

Zhe Ye, Fang Li, and Timothy Baldwin. 2018. Encoding sentiment information into word vectors for sentiment analysis. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 997–1007.

Liang-Chih Yu, Jin Wang, K. Robert Lai, and Xuejie Zhang. 2017. Refining word embeddings for sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing.*, pages 534 –539.

Evaluating Pretrained Transformer Models for Entity Linking in Task-Oriented Dialog

Sai Muralidhar Jayanthi, Varsha Embar, Karthik Raghunathan
MindMeld, Cisco Systems
{saijayan, vembar, ktick}@cisco.com

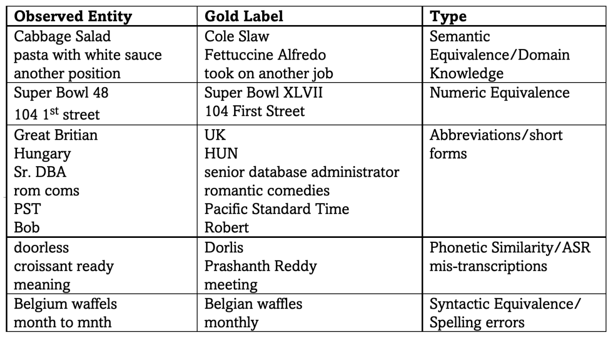
Abstract

The wide applicability of pretrained transformer models (PTMs) for natural language tasks is well demonstrated, but their ability to comprehend short phrases of text is less explored. To this end, we evaluate different PTMs from the lens of unsupervised Entity Linking in task-oriented dialog across 5 characteristics– *syntactic*, *semantic*, *short-forms*, *numeric* and *phonetic*. Our results demonstrate that several of the PTMs produce sub-par results when compared to traditional techniques, albeit competitive to other neural baselines. We find that some of their shortcomings can be addressed by using PTMs fine-tuned for text-similarity tasks, which illustrate an improved ability in comprehending semantic and syntactic correspondences, as well as some improvements for short-forms, numeric and phonetic variations in entity mentions. We perform qualitative analysis to understand nuances in their predictions and discuss scope for further improvements.[1]

1 Introduction

In task-oriented dialog systems, Entity Linking (EL) is the process of disambiguating a detected entity mention (*aka.* slot) in a user utterance to a canonical entry in a Knowledge Base (KB). EL is a crucial step in building robust dialog systems, especially when dealing with domain-specific entities, *e.g.*, a chatbot for food ordering or a voice assistant for medical assistance.

Popular open-source conversational AI platforms such as DeepPavlov (Burtsev et al., 2018), MindMeld (Raghuvanshi et al., 2018) and Rasa (Bocklisch et al., 2017) maintain a KB of canonical entries, each consisting of a title, optionally with aliases (*i.e.*, alternate usages) for the task of entity linking. Detected entities from user utterances, often with spelling and automatic speech recognition (ASR) errors, are then mapped to those canonical entries through text classification or similarity matching techniques.[2]



Observed Entity	Gold Label	Type
Cabbage Salad	Cole Slaw	Semantic Equivalence/Domain Knowledge
pasta with white sauce	Fettuccine Alfredo	
another position	took on another job	
Super Bowl 48	Super Bowl XLVII	Numeric Equivalence
104 1st street	104 First Street	
Great Britian	UK	Abbreviations/short forms
Hungary	HUN	
Sr. DBA	senior database administrator	
rom coms	romantic comedies	
PST	Pacific Standard Time	
Bob	Robert	
doorless	Dorlis	Phonetic Similarity/ASR mis-transcriptions
croissant ready	Prashanth Reddy	
meaning	meeting	
Belgium waffels	Belgian waffles	Syntactic Equivalence/ Spelling errors
month to mnth	monthly	

Figure 1: Different types of matching scenarios observed in Entity Linking task for short spoken/written language texts.

Previous works (Chen et al., 2020; Cao et al., 2021; Broscheit, 2019) have proposed *context-aware* classification techniques for EL, wherein the context surrounding the slots is leveraged to ascertain canonical names. However, such approaches fall short due to **(i)** their reliance on large training/fine-tuning sets and associated annotation costs **(ii)** requirement to re-train the classifiers with every change in KB entries. Alternatively, a more popular paradigm is to model EL as a matching problem by transforming entities into *vectors*, and using a similarity function such as cosine distance to find the closest canonical entry.

EL systems typically rely on textual n-gram features modeled by ranking algorithms such as BM25 (Robertson and Walker, 1994) implemented as part of search engines such as Elasticsearch.[3] To capture semantic similarity within such systems,

[1]Code and re-purposed datasets can be found at https://github.com/murali1996/el_tod

[2]Entity Linking may be clubbed with Entity Recognition or is a standalone component of the NLP pipeline, the latter is used in this work for better interpretability.

[3]https://www.elastic.co/blog/practical-bm25

one needs to tediously engineer feature sets and collect synonyms or aliases for each KB entry, leading to a lot of manual effort and development cost.

Recently, pretrained word embeddings have had much success in capturing entity correspondences (Francis-Landau et al., 2016; Sun et al., 2015) by addressing aforementioned shortcomings– off-the-shelf usage without reliance on training data and flexibility to expand KBs without retraining. Mudgal et al. (2018) presents a detailed account of different deep learning based representations and modeling choices for the EL task, showing the advantages of using them over traditional systems.

More recently, transformer-based PTMs like BERT (Devlin et al., 2019) have excelled for Entity Linking when entities are in the form of tabular data without much additional *context* (Tracz et al., 2020; Teong et al., 2020; Li et al., 2020; Mudgal et al., 2018). However, their ability to understand nuances in linking short spans of free-form text is not thoroughly tested, especially for domain-specific entities with minimal context.

In this work, we investigate and analyze how different PTMs behave in such settings, when compared to widely adopted neural and non-neural models (**§ 2**). To probe model behaviours on examples with different characteristics, we curate and benchmark evaluation datasets of various sizes that each contain a subset of those characteristics (**§ 3**). Lastly, we present qualitative as well as quantitative analysis of the predictions of various models, which shows that while pretrained models fine-tuned for text-similarity tasks perform the best overall, there is room for improvement (**§ 4**).

2 Models

In this section, we provide a brief detail of the different pretrained transformer models (PTMs) as well as the 5 baseline models (3 neural and 2 non-neural) used in our benchmarking process. We categorize PTMs under consideration into 4 different types to understand the usefulness of different pre-training strategies, number of parameters and inference times. We adopt the model nomenclature from Huggingface[4] (Wolf et al., 2020) and refer the reader to Rogers et al. (2020) and Qiu et al. (2020) for more comprehensive account on these different types of PTMs and their utility.

[4]https://huggingface.co/models

We categorize the PTMs as follows:

Type-I Pretrained general-purpose transformer language models which are *base*-sized. These include *bert-base-cased* (Devlin et al., 2019), *roberta-base* (Liu et al., 2019) and *mpnet-base* (Song et al., 2020).

Type-II Parameters-reduced models which are also trained for language modeling tasks through different parameter reduction techniques. These include *albert-base-v2* (Lan et al., 2020), *distilbert-base-cased* (Sanh et al., 2019), *distilroberta-base* (Sanh et al., 2019), and *MiniLM-L6-uncased* (Wang et al., 2020).

Type-III Reimers and Gurevych (2019) fine-tuned some of the Type-I and Type-II models on a variety of datasets annotated for textual similarity tasks[5]. We select their *all-** models which were fine-tuned with more than 1 billion textual pairs and were designed as general purpose textual similarity models. These include *all-distilroberta-v1*, *all-mpnet-base-v2* and *all-MiniLM-L6-v2*.

Type-IV Dynamic quantization can reduce the size of the model while only having a limited implication on accuracy. We use Pytorch's (Paszke et al., 2019) dynamic quantization functionality[6] to obtain the quantized versions of the following models: *all-mpnet-base-v2* and *all-MiniLM-L6-v2*.

In addition to the pretrained language models based on transformer architecture, we also benchmark PTMs based on other neural architectures. Specifically, we consider the following 3 neural models as baselines– (1) FASTTEXT (Bojanowski et al., 2017), (2) FLAIR (Akbik et al., 2019), and (3) ELMO (Peters et al., 2018).

FASTTEXT consists of continuous distributed word representations trained on large unlabeled corpora for many natural language processing tasks. It represents each word as the sum of its character n-grams. Compared to FLAIR and ELMO, this model has a shallower network and is pretrained similar to Mikolov et al. (2013)'s skipgram model with negative sampling. In our benchmarking, we use the 300-dimension English model.[7]

[5]https://www.sbert.net/docs/pretrained_models.html
[6]https://pytorch.org/dynamic_quantization_bert_tutorial.html
[7]https://github.com/facebookresearch/fastText/crawl-vectors.md

FLAIR is a LSTM based pretrained character language model (Hochreiter and Schmidhuber, 1997), trained to produce a novel type of word embedding also known as *contextual string embeddings*. It is trained without any explicit notion of words and hence can represent even out-of-vocabulary (OOV) words similar to FASTTEXT. In our experiments, we use word representations concatenated from their *news-forward* and *news-backward* models leading to 4096-dimensional vectors.[8]

ELMO is a deep contextualized bidirectional word representation produced by pretrained LSTMs. In our experiments, we use the *base* model and concatenate all three ELMo layers leading to 3072-dimensional vectors.[9]

We compare all the above neural models with two non-neural baselines which are popularly adopted for the task at hand– (1) TFIDF vectorizer[10] and (2) BM25, both using word & character n-grams upto 5-gram.

For all models except BM25, we use cosine similarity as the scoring function. For every pretrained model, we use mean pooled representation of all (sub-)words in a given entity text as its final representation.[11]

3 Datasets

We utilize both in-house and publicly available corpora to curate datasets in English for the Entity Linking task– MindMeld Blueprints dataset[12](MM_BP) along with word and character level misspelled versions of this data (MM_BP-WORD and MM_BP-CHAR), re-purposed open-domain QA datasets like ComplexWebQuestions (COMPLWQ) (Talmor and Berant, 2018) and MKQA (MKQA) (Longpre et al., 2020), acronym identification dataset (ACRI) (Veyseh et al., 2020), and an in-house dataset of ASR mis-transcriptions for person names (ASR-MIS). More details on the dataset curation process is provided in Appendix A.

To probe model behaviours further, we manually annotate 1.3K queries pooled from all of these datasets into our 5 predefined categories as follows (with their sample sizes) – SEMANTIC (#294), SYNTACTIC (#408), SHORT-FORMS (#310), NUMERALS (#125) and PHONETICS (#200). Examples from these sets are presented in Figure 1.

We use Precision@1 (P@1) and Precision@5 (P@5) as our benchmarking metrics and conduct all our experiments using the publicly available MindMeld framework[13]. Unless otherwise stated, we do not include any aliases alongside canonical titles for matching KB entries and utilize all known aliases as our test queries. We disregard any canonical descriptions as they are not always available and procuring them may have significant annotation costs.

4 Results & Analysis

Table 1 present the results of different models across our curated datasets. We observe that on average, Type-I & Type-II models perform poorly compared to the baselines by atleast 30% P@1. However, Type-III & Type-IV models, fine-tuned to find similar sentence pairs, perform superior to our baselines by 5-13%, showcasing the usefulness of such tuning strategies even to short texts. We further observe that the parameter-reduced models generally perform better than the *base* models. Almost all PTMs perform poorly on abbreviations and also fail to beat the BM25 baseline on the phonetic matching dataset. While we believe that these two datasets are quite challenging to the PTMs as their training processes do not include any related objectives, the superior performance of Type-III models compared to Type-I and Type-II is quite encouraging. On misspelled versions of the datasets, Type-III & Type-IV models still perform better than others. However, their precision falls short by at least 10% absolute indicating scope for improvement.

4.1 Qualitative analysis

Figure 2 shows the performance of different models on the 5 different categories of data without and with aliases in the KB. We perform a manual inspection of the results across the 5 categories with 3 different models: baseline BM25 model, Type-I *bert-base-cased* (BERT) and Type-IV *all-mpnet-base-v2-quantized* (MPNET-Q).

[8] https://github.com/flairNLP/flair/FLAIR_EMBEDDINGS.md

[9] https://github.com/allenai/bilm-tf

[10] https://scikit-learn.org/sklearn-TFIDF

[11] Different pretrained models have different tokenization strategies and we leave any analysis on the effect of tokenization to future work.

[12] https://github.com/CiscoDevNet/mindmeld-blueprints

[13] https://github.com/cisco/mindmeld

Results for Entity Linking (Precision@1 / Precision@5)

		MM_BP	COMPLWQ	MKQA	ACRI	ASR-MIS	*Avg.*	MM_BP-WORD *before*	MM_BP-WORD *after*	MM_BP-CHAR *before*	MM_BP-CHAR *after*
Baselines	BM25	49.5 / 52.6	55.1 / 66.0	54.2 / 59.6	1.1 / 1.5	**52.3** / 66.0	42.4 / 49.2	41.9 / 54.9	33.5 / 41.0	46.5 / 60.8	**37.6** / 48.9
	TFIDF	66.7 / 88.3	55.7 / 69.9	67.3 / 84.0	1.0 / 2.1	39.4 / **66.3**	46.0 / 62.1	41.1 / 87.5	**35.3** / 86.3	43.3 / 87.6	36.2 / **87.7**
	FLAIR	44.1 / 81.7	32.8 / 40.2	19.7 / 23.8	0.1 / 0.3	12.9 / 20	21.9 / 33.2	17.1 / 81.9	11.4 / 79.4	20.4 / 82.3	12.1 / 75.3
	FASTTEXT	60.9 / 89.8	37.1 / 47.9	24.4 / 30.7	6.0 / 11.5	4.0 / 7.4	26.5 / 37.5	26.0 / 87.4	13.5 / 82.4	29.4 / 88.7	13.5 / 76.1
	ELMO	57.7 / 84.6	46.6 / 58.3	26.6 / 32.6	1.1 / 1.9	6.4 / 10.4	27.7 / 37.6	17.5 / 86.3	10.4 / 81.5	21.6 / 83.3	10.9 / 75.1
Type-I	*bert-base-cased*	45.9 / 79.4	40.2 / 49.2	27.2 / 34.3	0.6 / 1.2	5.7 / 8.9	23.9 / 34.6	15.4 / 82.0	8.3 / 75.8	18.9 / 78.2	9.6 / 73.7
	roberta-base	59.0 / 76.7	43.9 / 40.4	38.5 / 30.7	1.7 / 0.9	13.1 / 9.3	31.2 / 31.6	27.4 / 78.8	13.5 / 75.3	29.4 / 77.1	17.2 / 72.3
	mpnet-base	31.1 / 74.3	25.2 / 30.1	21.4 / 23.0	0.4 / 0.6	12.3 / 14.0	18.1 / 28.4	21.9 / 77.6	7.1 / 73.1	22.2 / 74.2	8.3 / 69.2
Type-II	*albert-base-v2*	39.1 / 77.8	22.9 / 28.5	17.4 / 19.2	0.1 / 0.3	5.8 / 7.8	17.1 / 26.7	18.2 / 79.9	6.6 / 74.4	18.1 / 78.3	7.0 / 71.7
	distilbert-base-cased	52.8 / 81.1	29.5 / 33.3	37.7 / 41.0	0.6 / 0.7	6.5 / 9.5	25.4 / 33.1	15.4 / 83.9	9.9 / 74.6	19.0 / 80.4	11.1 / 72.8
	distilroberta-base	64.6 / 77.7	39.6 / 31.9	36.1 / 26.5	1.5 / 0.7	10.6 / 6.5	30.5 / 28.7	25.0 / 80.8	14.0 / 74.2	26.2 / 76.9	16.3 / 73.3
	MiniLM-L6-uncased	57.5 / 82.7	33.7 / 38.0	36.8 / 38.9	0.5 / 0.8	16.9 / 18.3	29.1 / 35.8	29.5 / 85.6	9.8 / 75.6	30.6 / 81.4	11.4 / 73.8
Type-III	*all-distilroberta-v1*	72.3 / 91.8	62.5 / 72.1	59.9 / 76.4	**11.2 / 23.3**	42.4 / 62.6	49.7 / 65.2	42.2 / 91.7	24.2 / **87.3**	44.3 / 91.4	32.2 / 87.5
	all-mpnet-base-v2	75.8 / 91.4	62.3 / 72.1	57.6 / 73.3	6.0 / 11.4	43.6 / 57.5	49.1 / 61.1	44.8 / **92.4**	21.6 / 85.4	46.3 / 90.9	27.5 / 86.3
	all-MiniLM-L6-v2	74.6 / 91.7	62.0 / 71.8	68.2 / 80.1	4.7 / 8.2	44.9 / 59.2	50.9 / 62.2	**45.5** / 91.4	21.5 / 82.9	47.5 / 92.0	27.4 / 84.8
Type-IV	*all-mpnet-base-v2* (Q)	**79.4 / 93.3**	**63.2 / 72.4**	**75.5 / 84.1**	5.5 / 10.6	43.5 / 59.5	**53.4 / 64.0**	**45.5 / 92.4**	22.1 / 85.4	46.7 /**92.5**	28.4 / 85.8
	all-MiniLM-L6-v2 (Q)	73.0 / 91.0	61.2 / 70.9	67.3 / 79.6	4.1 / 7.4	42.3 / 58.1	49.6 / 61.4	44.8 / 91.3	20.9 / 83.0	**46.8** / 90.9	26.5 / 82.3

Table 1: Evaluation of different pretrained transformer models across different datasets (§ 3). The *Avg.* column reports mean precision across different datasets. Marked in **bold** are the best scores & in underline are second best.

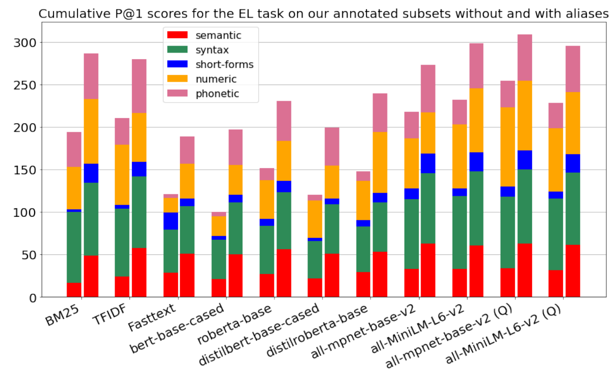


Figure 2: EL results on annotated subset of 1.3K test queries, annotated across 5 matching criterion. For each model, the first bar corresponds to the scenario with KBs containing only canonical names whereas for the second, KBs contain aliases in addition for disambiguating test queries.

4.1.1 Syntactic Matches

Syntactic matches refer to cases when the query and its matching canonical form have slight textual variations or spelling errors. The baseline TFIDF and BM25 models are well equipped to handle such differences and perform on-par and in some cases, better than the other models. Between the BERT and MPNET-Q models, the latter handles syntactic differences better than the former by favouring more word overlaps.

```
Query: John Jr.
BM25: John F. Kennedy Jr.
BERT: Michael Joseph Jackson, Jr.
MPNET-Q: John Warner

Query: mammoth pizza
BM25: Wham, Bam, Thank You Mammoth
BERT: Pizzawich
MPNET-Q: Fresco Pizza

Query: Hindi
BM25: Hindi Language
BERT: India
MPNET-Q: Hindi Language
```

4.1.2 Semantic Matches

The baseline BM25 system relies heavily on aliases to handle queries that are semantically equivalent to one of the canonical names in the KB. In their absence, the model performs poorly in this category. In contrast, the transformer models are better suited to handle these queries. We notice 2 trends in the BERT and MPNET-Q models:
While BERT tends to predict related words, they are not always semantically equivalent.

```
Query: Instrumentalist
BERT: Singer
MPNET-Q: Musician

Query: most recently released
BERT: popular
MPNET-Q: latest

Query: totalled
BERT: count
MPNET-Q: sum
```

In addition, the BERT system tends to rank antonyms higher.

```
Query: min
BERT: highest
MPNET-Q: lowest

Query: hilarious
BERT: erotic
MPNET-Q: comedy

Query: resigned
BERT: active
MPNET-Q: voluntarily terminated
```

4.1.3 Abbreviations & Short Forms

All models perform poorly on abbreviations and short forms. BM25 relies on character n-grams to match shortened sub-strings of entities, but fails on acronyms. MPNET-Q is able to identify acronyms of popular entities like universities, countries, etc., perhaps as a result of the fine-tuning phase.

```
Query: PSU Football
BM25: Football
BERT: UD Arena
MPNET-Q: Penn State Nittany Lions
         football

Query: Mla
BM25: Mlabri Language
BERT: lo
MPNET-Q: Mlabri Language

Query: USSR
BM25: (no result)
BERT: Czechoslovakia
MPNET-Q: Soviet Union
```

4.1.4 Numeric Matches

Among the three systems, BERT performs the worst with numeric entities. It does not handle different numeric representations of the same entity well, leading to random predictions. Fuzzy character matching ensures that BM25 system handles different formats well as long as most of the characters match. MPNET-Q model handles changes in numeric formats the best even when compared against its full model, with P@1 of 92.8.

```
Query: 90's
BM25: 1990s
BERT: 2010s
MPNET-Q: 1990s

Query: 5th Avenue
BM25: 12th Avenue
BERT: 12th Avenue
MPNET-Q: 45 Fifth Avenue

Query: 1775 April 19
BM25: april 1986
BERT: 1875-09
MPNET-Q: 1775-04-19
```

4.1.5 Phonetic Matches

Often, ASR systems mis-transcribe uncommon words into more common, phonetically similar words. This category tests whether the models are robust to such errors. While the performance of all the models are lacking, BM25 qualitatively provides explainable results due to its reliance on textual similarities when compared to predictions of the PTMs. Typically, EL systems are evaluated on queries that test the models' abilities to match the 4 categories mentioned above. Given the popularity of conversational agents with a speech interface, probing EL models for their phonetic matching capabilities is important.

```
Query: this loud (Liz Laub)
BM25: Cloud Hu
BERT: Kevin Upright
MPNET-Q: Riley Rant

Query: Yale sushi (Xiaoxue Shi)
BM25: Sakshi Alekar
BERT: Joshua Frattarola
MPNET-Q: Sammy Su
```

5 Conclusion

Given the success of PTMs for various NLP applications (Rogers et al., 2020), we evaluate the ability of these models to understand short spans of text for unsupervised entity linking in task-oriented dialog systems by curating a large dataset and comparing their results against traditional n-gram systems. We further analyze the performance of these models across 5 different characteristics-syntactic, semantic, abbreviations & short-forms, numeric and phonetic matches. Our results demonstrate that these models, when fine-tuned on a semantic similarity task, comprehend syntactic and semantic differences in short phrases better than their other variants. However, their performance is lacking - particularly for abbreviations and queries with speech recognition errors - with the best performing models averaging at 53.4% P@1 and 64.0% P@5 across the different datasets. For future work, with the goal of creating a generic model for the unsupervised EL task, we plan to improve these models through task-adaptive fine-tuning techniques with our curated datasets.

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. Flair: An easy-to-use framework for state-of-the-art nlp. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.

Tom Bocklisch, Joe Faulkner, Nick Pawlowski, and Alan Nichol. 2017. Rasa: Open source language understanding and dialogue management. *ArXiv*, abs/1712.05181.

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.

Samuel Broscheit. 2019. Investigating entity knowledge in BERT with simple neural end-to-end entity linking. In *Proceedings of the 23rd Conference on*

Computational Natural Language Learning (CoNLL), pages 677–685, Hong Kong, China. Association for Computational Linguistics.

Mikhail Burtsev, Alexander Seliverstov, Rafael Airapetyan, Mikhail Arkhipov, Dilyara Baymurzina, Nickolay Bushkov, Olga Gureenkova, Taras Khakhulin, Yuri Kuratov, Denis Kuznetsov, Alexey Litinsky, Varvara Logacheva, Alexey Lymar, Valentin Malykh, Maxim Petrov, Vadim Polulyakh, Leonid Pugachev, Alexey Sorokin, Maria Vikhreva, and Marat Zaynutdinov. 2018. DeepPavlov: Open-source library for dialogue systems. In *Proceedings of ACL 2018, System Demonstrations*, pages 122–127, Melbourne, Australia. Association for Computational Linguistics.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *International Conference on Learning Representations*.

Haotian Chen, Xi Li, Andrej Zukov Gregoric, and Sahil Wadhwa. 2020. Contextualized end-to-end neural entity linking. 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 637–642, Suzhou, China. 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.

Matthew Francis-Landau, Greg Durrett, and Dan Klein. 2016. Capturing semantic similarity for entity linking with convolutional neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1256–1261, San Diego, California. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

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 *International Conference on Learning Representations*.

Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep entity matching with pre-trained language models. *Proc. VLDB Endow.*, 14(1):50–60.

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*, abs/1907.11692.

Shayne Longpre, Yi Lu, and Joachim Daiber. 2020. Mkqa: A linguistically diverse benchmark for multilingual open domain question answering.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Sidharth Mudgal, Han Li, Theodoros Rekatsinas, A. Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and V. Raghavendra. 2018. Deep learning for entity matching: A design space exploration. *Proceedings of the 2018 International Conference on Management of Data*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. 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 8024–8035. Curran Associates, Inc.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *ArXiv*, abs/2003.08271.

Arushi Raghuvanshi, Lucien Carroll, and Karthik Raghunathan. 2018. Developing production-level conversational interfaces with shallow semantic parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 157–162, Brussels, Belgium. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

S. E. Robertson and S. Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM*

SIGIR Conference on Research and Development in Information Retrieval, SIGIR '94, page 232–241, Berlin, Heidelberg. Springer-Verlag.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. *ArXiv*, abs/2004.09297.

Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling mention, context and entity with neural networks for entity disambiguation. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 1333–1339. AAAI Press.

A. Talmor and J. Berant. 2018. The web as a knowledge-base for answering complex questions. In *North American Association for Computational Linguistics (NAACL)*.

Kai-Sheng Teong, Lay-Ki Soon, and Tin Tin Su. 2020. *Schema-Agnostic Entity Matching Using Pre-Trained Language Models*, page 2241–2244. Association for Computing Machinery, New York, NY, USA.

Janusz Tracz, Piotr Wójcik, Kalina Jasinska-Kobus, Riccardo Belluzzo, Robert Mroczkowski, and Ireneusz Gawlik. 2020. Bert-based similarity learning for product matching. In *ECOMNLP*.

Amir Pouran Ben Veyseh, Franck Dernoncourt, Quan Hung Tran, and Thien Huu Nguyen. 2020. What Does This Acronym Mean? Introducing a New Dataset for Acronym Identification and Disambiguation. In *Proceedings of COLING*.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*, volume 33, pages 5776–5788. Curran Associates, Inc.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Cascading Adaptors to Leverage English Data to Improve Performance of Question Answering for Low-Resource Languages

Hariom A. Pandya [*] [†]
Dharmsinh Desai University
Nadiad-Gujarat(India)
pandya.hariom@gmail.com

Bhavik Ardeshna [*] [†]
Dharmsinh Desai University
Nadiad-Gujarat(India)
ardeshnabhavik@gmail.com

Brijesh S. Bhatt
Dharmsinh Desai University
Nadiad-Gujarat(India)
brij.ce@ddu.ac.in

Abstract

Transformer based architectures have shown notable results on many down streaming tasks including question answering. The availability of data, on the other hand, impedes obtaining legitimate performance for low-resource languages. In this paper, we investigate the applicability of pre-trained multilingual models to improve the performance of question answering in low-resource languages. We tested four combinations of language and task adapters using multilingual transformer architectures on seven languages similar to MLQA dataset. Additionally, we have also proposed zero-shot transfer learning of low-resource question answering using language and task adapters. We observed that stacking the language and the task adapters improves the multilingual transformer models' performance significantly for low-resource languages. Our code and trained models are available at : https://github.com/CALEDIPQALL/

1 Introduction

Last few years have seen emergence of transformer based pretrained models like BERT(Devlin et al., 2019), XLNet(Yang et al., 2019), T5(Raffel et al., 2020), XLM-RoBERTa(Conneau et al., 2020) etc. The pretrained models have shown significant improvement in various downstream tasks like question answering, NER, Machine translation and speech recognition(Delobelle et al., 2020; Pires et al., 2019; Pfeiffer et al., 2020a; Pires et al., 2019; Pandya and Bhatt, 2021; Saha et al., 2021; Murthy et al., 2019; Park et al., 2008; Raffel et al., 2020).

The emergence of multilingual models: mBERT (Devlin et al., 2019) and XLM-RoBERTa(Conneau et al., 2020) made it possible to leverage English data to improve the performance of low-resource languages. In this paper, we continue to investigate the effectiveness of multilingual pretrained transformer models in improving the performance of question answering systems in a low-resource setup using the cascading of language and task adapters(Pfeiffer et al., 2021, 2020a; Bapna and Firat, 2019). Our work contributes by evaluating cross-lingual performance in seven languages - Hindi, Arabic, German, Spanish, English, Vietnamese and Simplified Chinese. Our models are evaluated on the combination of XQuAD(Artetxe et al., 2020) and MLQA(Lewis et al., 2020) datasets which are similar to SQuAD (Rajpurkar et al., 2016) .

To this end, our contributions are as follows:

- We have trained multilingual variants of transformers, namely mBert and XLM-RoBERTa with a QA dataset in seven languages. Both the MLQA and XQuAD datasets contain validation and test sets for the above languages but not the training set. To finetune the model we have combined the test set of XQuAD and MLQA datasets and evaluated the model with the MLQA development dataset as the test dataset. By splitting the dataset in this way we can get train and test data with the considerable length for low-resource languages which helped us to conduct various experiments. Table 1 highlights the size of our train and test set for all the above-mentioned languages.
- We exhaustively analysed the fine-tuned models by evaluating them with the tasks adapter[1] (Pfeiffer et al., 2021, 2020a). We conducted the experiments in two different setups, Houlsby(Houlsby et al., 2019) and Pfeif-

[*]Equal contribution
[†]Corresponding Authors

[1]Pre-trained task adapters from https://adapterhub.ml/explore/qa/squad1/

	Hindi	German	Spanish	Arabic	Chinese	Vietnamese	English
Train	6854	5707	6443	6525	6327	6685	12780
Test	507	512	500	517	504	511	1148

Table 1: Size of the train and test set used in the experiments. The MLQA(Lewis et al., 2020) test and XQuAD(Artetxe et al., 2020) datasets are used for fine-tuning the model and for testing purpose MLQA devset is used for all languages to maintain consistency.

fer(Pfeiffer et al., 2021, 2020b). These two setups enabled us to compare our language model variants with their multilingual counterparts and understand the different factors that lead to better results on the downstream tasks.

- We have also attempted a series of two different experiments by stacking language adapters and task adapter[2] in different ways. We first analyze the fine-tuned model by stacking language-specific adapter with the XLM-RoBERTa$_{base}$ [3]. After fine-tuning the language-specific adapter we augment the task-specific adapter upon the previously fine-tuned language adapter. We analyze both the experiments separately and conclude that multiple adapters with the transformer-based model perform notably better.

- Due to limited training, the transfer-learning performance of the transformer is poor on the low-resource languages as well as on the languages unseen during the pretraining(Kakwani et al., 2020). The multi-task adapter (MAD-X) (Pfeiffer et al., 2020b) outperforms the state-of-the-art models in cross-lingual transfer across a representative set of typologically diverse languages on question answering. To avoid the training of model individually for multiple languages while maintaining the performance, we used cross-lingual transfer by switching heads of language adapter from the source language to the target language.

2 Proposed Approach

In this section we describe our approach of training the task adapter and the language adapters in 4 different setup.

2.1 Cross-Lingual Tuning of Task Adapter and Language Adapters

Task-Specific Cross-Lingual Transfer: We have used two different configurations for fine-tuning the task-specific adapter for cross-lingual transfer in low-resource languages (Pfeiffer et al., 2021; Houlsby et al., 2019). We have fine-tuned XLM-RoBERTa$_{base}$ for multiple languages with the question answering corpora. We calculated the F1-Score, Exact Match, Jaccard [4] , and WER (Word Error Rate)(Park et al., 2008) [5] for the test dataset.

Adapting Cross-Lingual learning using Language-Specific Model: We used the language adapter trained using unlabelled data on MLM objective. It makes the pretrained multilingual model more suitable for the specific language with its improved language understanding. We perform the downstream task by stacking specific language adapter with the XLM-RoBERTa$_{base}$ and used recent efficient adapter architecture proposed by pfeiffer et al. (Pfeiffer et al., 2021).

After fine-tuning task-specific adapter and language-specific adapters individually with the different low-resource languages, we observed that by stacking task adapter and language adapters together with the transformer model the performance improved significantly. For each language available in MLQA, we fine-tuned a task adapter using a corresponding question answering dataset.

2.2 Multi-Task Adapter for Cross-Lingual Transfer

The adapter-based MAD-X framework (Pfeiffer et al., 2020b) enables learning language-specific and task-specific transformations in a modular and parameter-efficient way. Our method of using MAD-X is as follows:

[2]Pre-Trained Language Adapters from `https://adapterhub.ml/explore/text_lang/`

[3]XLM-RoBERTa$_{base}$ `https://huggingface.co/deepset/roberta-base-squad2`

[4]Jaccard score `https://en.wikipedia.org/wiki/Jaccard_index`

[5]WER score `https://en.wikipedia.org/wiki/Word_error_rate`

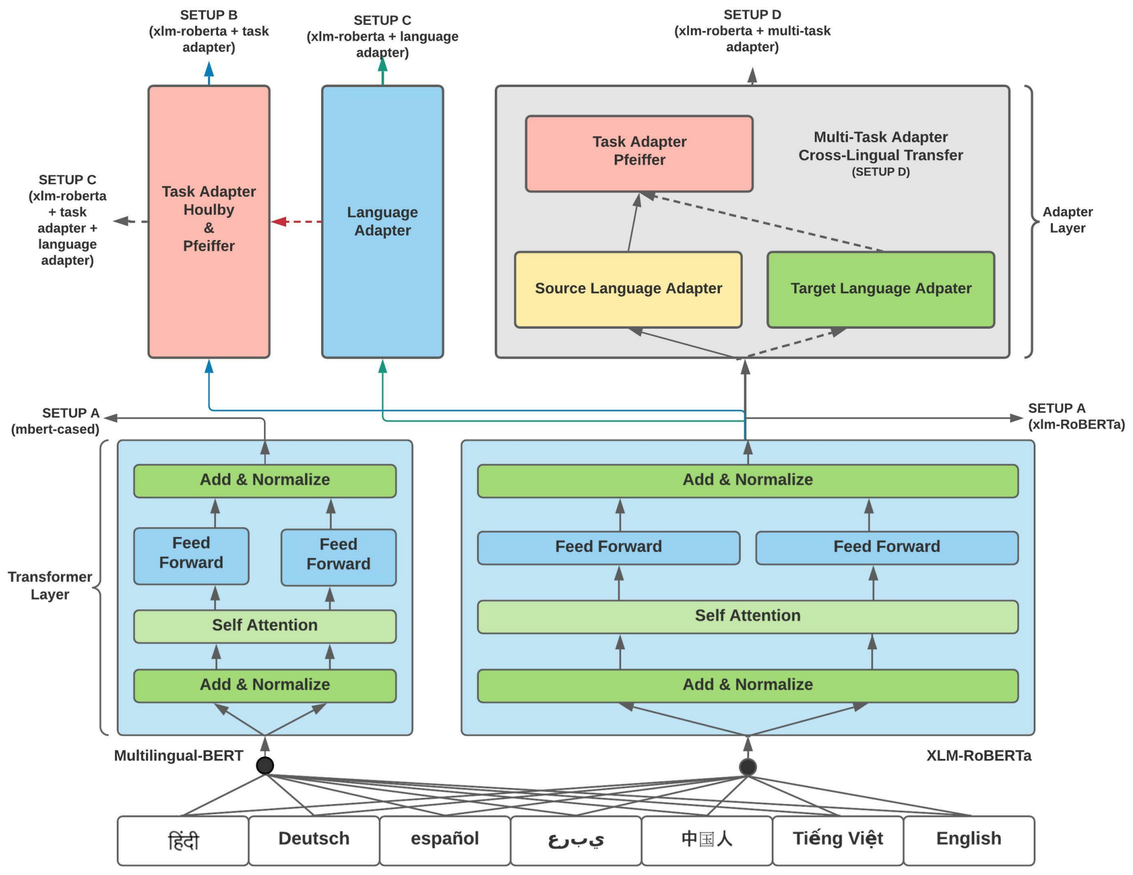


Figure 1: Experimental architecture- SETUP A: mBERT and XLM-R for QA, SETUP B: XLM-R with task adapters setup, SETUP C: XLM-R with language and language + task adapter and SETUP D: MAD-X setup for XLM-R

1. We have used pre-trained language adapters[6] for the source and target language on a language modeling task.

2. Train a task adapter on the target task dataset. This task adapter is stacked upon the previously trained language adapter. During this step, only the weights of the task adapter are updated.

3. Next, in zero-shot cross-lingual transfer step, we replaced the source language adapter with the target language adapter while keeping the stacked task adapter.

3 Experimental setups

We have performed 4 different analysis as represented in Figure 1. Details of all 4 setups are shown below:

[6]from https://adapterhub.ml/

3.1 Setup A

Here, we evaluated mBERT, XLM-Roberta$_{base}$ and XLM-Roberta$_{large}$ models on downstream tasks with the training dataset, which is specific to the individual language variant. The EM and F1 score for all languages are shown in Table 2.

Here, the interpretation of the matrix is F1/EM and it is same for rest of the Setups. For Example, in Table 2 first entry 56.25/39.45 indicates, for the Hindi test set, the F1score=56.25 and EM=39.45 is achieved using mBERT transformer model.

3.2 Setup B

After fine-tuning the transformer model, We have evaluated XLM-RoBERTa$_{base}$ with the task-specific adapter on downstream tasks under two training settings: Houlby(Houlsby et al., 2019) and Pfeiffer(Pfeiffer et al., 2021). While fine-tuning, the weights of only the task adapter get updated and the model weights are kept unchanged. This setup enables the scalable sharing of the task adapter model particularly in low-resource scenarios. Pre-

	Hindi	**German**	**Spanish**	**Arabic**	**Chinese**	**Vietnamese**	**English**
mBERT	56.25 / 39.45	52.99 / 38.09	59.89 / 40.4	51.28 / 31.33	41.86 / 41.07	59.52 / 39.73	77.86 / 63.85
XLM-RoBERTa$_{base}$	64.49 / 48.32	60.74 / 45.31	68.99 / 47.6	58.07 / 39.65	45.37 / 44.24	68.19 / 48.53	81.29 / 68.64
XLM-RoBERTa$_{large}$	73.37 / 56.02	70.57 / 53.32	76.32 / 54.2	67.15 / 47.78	49.94 / 49.21	73.78 / 54.21	85.98 / 74.39

Table 2: F1 score and Exact Match on the test set for the Setup A on multilingual-BERT and XLM-RoBERTa.

	Hindi	**German**	**Spanish**	**Arabic**	**Chinese**	**Vietnamese**	**English**
Task Adapter (*Houlby*)	64.12 / 47.73	**60.95 / 44.53**	68.48 / 46.6	**58.13 / 38.49**	**44.38 / 43.25**	68.39 / 48.34	80.86 / 68.29
Task Adapter (*Pfeiffer*)	**65.7 / 49.9**	60.53 / 44.14	**69.09 / 48**	55.97 / 37.14	44.05 / 43.05	**68.46 / 48.53**	**81.23 / 68.64**

Table 3: F1 score and Exact Match for the xlm-roberta with Task Adapter (Setup B). We bold the best results.

trained task-specific adapters: Houlby[7] and Pfeiffer[8] are taken with predefined conditions. The EM and F1 score for all languages are shown in Table 3.

3.3 Setup C

The language adapters are used to learn language-specific transformations (Pfeiffer et al., 2020b). After being trained on a language modeling task, a language adapter can be stacked before a task adapter for training on a downstream task. To perform zero-shot cross-lingual transfer, one language adapter can be replaced by another. In terms of architecture, language adapters are largely similar to task adapters, except for an additional invertible adapter layer after the embedding layer.

In this setup, we have evaluated each language-specific adapter[9] by stacking it on the XLM-RoBERTa model. In the second phase, we stacked the task-specific adapter and language-specific adapter on the XLM-RoBERTa model. The EM and F1 score for the language adapter and the task + language adapter fusion are shown in Table 4.

3.4 Setup D

Here, we have cascaded the multi-task adapters(Pfeiffer et al., 2020b) to leverage the high-resource dataset to improve the performance of the low-resource language. We stacked the fine-tuned task-specific adapter upon the language-specific adapter and XLM-RoBERTa (shown in figure 1). After fine-tuning with high resource language, we performed zero-shot cross-lingual transfer by switching the source language adapter with the target language adapters.

[7]Available at https://adapterhub.ml/adapters/ukp/roberta-base_qa_squad1_houlsby/

[8]https://adapterhub.ml/adapters/ukp/roberta-base_qa_squad1_pfeiffer/

[9]Available at https://adapterhub.ml/explore/text_lang/

Our results for multi-task adapters are highlighted in the Table 6.

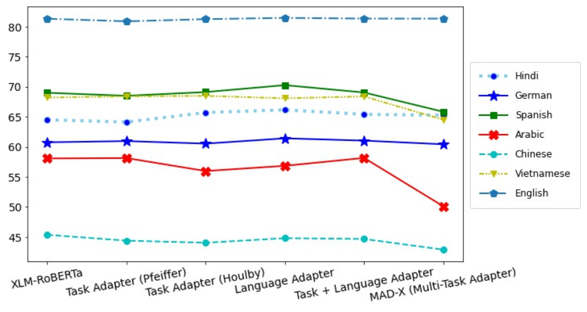


Figure 2: The performance of the different heads. The Y-axis here denotes the F1 score

Table 5 shows Jaccard and WER score for all four setups while the Figure 2 represents the F1 score of our models on all the languages.

4 Observations

To study the impact of the task adapter and the language adapters, we have conducted experiments as shown in Setup B and Setup C. Our observations from Table 3 and Table 4 indicates that the trained language adapter (Setup C: language adapters only) improves the performance for Hindi, German, Spanish, Chinese and English languages over the usage of task adapter(Setup B). However, instead of using language adapters only the stack of task and the language adapters lower EM and F1 score for languages other than Arabic.

We have compared two task adapter architectures and noted that the usage of different task adapter architectures have negligible performance impact on majority of the languages. As a result, no clear distinction can be drawn from this observation, which can be used to guide future research.

High-resource languages that use the Right-to-Left (RTL) scripting approach dominate the training of pretrained transformer models. The Arabic language follows Left-to-right(LTR) scripting

	Hindi	German	Spanish	Arabic	Chinese	Vietnamese	English
Language Adapter	**66.14 / 49.11**	**61.41 / 45.9**	**70.25 / 49.6**	56.84 / 37.52	**44.82 / 43.85**	68.06 / **49.31**	**81.43** / 68.64
Task + Language Adapter	65.39 / 48.72	61.03 / 45.51	69.03 / 47.6	**58.15 / 38.29**	44.68 / 43.45	**68.39** / 48.14	81.31 / **68.9**

Table 4: F1 score and Exact Match on the test set for the Setup C and We bold the best result in each section.

	Hindi	German	Spanish	Arabic	Chinese	Vietnamese	English
XLM-RoBERTa$_{base}$	59.1 / 76.9	51 / 94.9	53 / **74.4**	**50 / 92.7**	**44.8** / 60.2	57.2 / 81.6	73.6 / 49.3
Task Adapter (*Pfeiffer*)	58.2 / 84.7	51 / **93**	52.2 / 88.9	49 / 92.8	43.9 / **59.2**	**57.3** / 81	73.4 / 50.7
Task Adapter (*Houlby*)	59.7 / **70.6**	51.1 / 93.4	53.1 / 78.9	48.5 / 87.6	43.6 / 60.1	57.1 / 79.9	73.6 / 49.7
Language Adapter	**60.4** / 74.7	**52.5** / 104.2	**54.6** / 75.9	49.2 / 93.8	44.3 / 59.7	56.6 / **76.4**	**73.8** / 49.4
Task + Language Adapter	59.5 / 82.8	51.6 / 97.4	53 / 76.9	49.8 / 93.7	44.1 / 59.4	57.2 / 85.2	73.7 / **46.5**
MAD-X (Multi-Task Adapter)	59.7 / 72.6	48.6 / 95.5	50.3 / 88.7	42.9 / 107	42.4 / 60.9	53.7 / 88.4	-

Table 5: Jaccard and Word error rate (WER) on the test set for Setup A, B, C, and D

	Multi-Task Adapter (Task + Source Language + Target Language)
Hindi	65.24 / 48.91
German	60.42 / 43.35
Spanish	65.82 / 44.2
Arabic	50.12 / 31.33
Chinese	42.87 / 41.86
Vietnamese	64.48 / 44.22
English	-

Table 6: F1 score and Exact Match on the test set for the Setup D of Multi-Task adapter

style. The general poor performance in the Arabic language could be due to a variation in scripting technique. This also demonstrates that, regardless of the downstream task, the language structure has a significant impact on overall performance.

The Chinese language has a symbolic language structure and can be written in a variety of forms (left-to-right, or vertically top-to-bottom). The degraded findings in Chinese compared to other low-resource languages are most likely due to the language's writing flexibility.

Acknowledgments

The PARAM Shavak HPC computer facility is used for some of our experiments. We are grateful to the Gujarat Council of Science and Technology (GUJCOST) for providing this facility to the institution so that deep learning studies are being carried out effectively.

5 Conclusions

We have investigated the efficacy of cascading adapters with transformer models to leverage high-resource language to improve the performance of low-resource languages on the question answering task. We trained four variants of adapter combinations for - Hindi, Arabic, German, Spanish, English, Vietnamese, and Simplified Chinese languages. We demonstrated that by using the transformer model with the multi-task adapters, the performance can be improved for the downstream task. Our results and analysis provide new insights into the generalization abilities of multilingual models for cross-lingual transfer on question answering tasks.

References

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online. Association for Computational Linguistics.

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. 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 1538–1548, Hong Kong, China. 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.

Pieter Delobelle, Thomas Winters, and Bettina Berendt. 2020. RobBERT: a Dutch RoBERTa-based Language Model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3255–3265, Online. 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.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

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 the Association for Computational Linguistics: EMNLP 2020*, pages 4948–4961, Online. Association for Computational Linguistics.

Patrick Lewis, Barlas Oguz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. 2020. MLQA: Evaluating cross-lingual extractive question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7315–7330, Online. Association for Computational Linguistics.

Rudra Murthy, Anoop Kunchukuttan, and Pushpak Bhattacharyya. 2019. Addressing word-order divergence in multilingual neural machine translation for extremely low resource languages. 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 3868–3873, Minneapolis, Minnesota. Association for Computational Linguistics.

HA Pandya and BS Bhatt. 2021. Question answering survey: Directions, challenges, datasets, evaluation matrices. *Journal of Xidian University*, 15(4):152–168.

Youngja Park, Siddharth Patwardhan, Karthik Visweswariah, and Stephen Gates. 2008. An empirical analysis of word error rate and keyword error rate. pages 2070–2073.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. AdapterHub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Tulika Saha, Dhawal Gupta, Sriparna Saha, and Pushpak Bhattacharyya. 2021. A unified dialogue management strategy for multi-intent dialogue conversations in multiple languages. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 20(6).

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

eaVQA: An Experimental Analysis on Visual Question Answering Models

Souvik Chowdhury[1] and **Badal Soni**[1]
[1]National Institute of Technology, Silchar
souvik21_rs@cse.nits.ac.in, badal@cse.nits.ac.in

Abstract

Visual Question Answering (VQA) has recently become a popular research area. VQA problem lies in the boundary of Computer Vision and Natural Language Processing research domains.Various details about each dataset are given in this paper, which can help future researchers to a great extent. In this paper, we discussed and compared the experimental performance of the Stacked Attention Network Model (SANM) and bidirectional Long Short Term Memory (LSTM) and Multimodal Tucker Fusion (MUTAN) based fusion models. As per the experimental results, MUTAN accuracy and loss are 29% and 3.5, respectively. SANM model is giving 55% accuracy and a loss of 2.2, whereas the VQA model is giving 59% accuracy and 1.9 loss.

1 Introduction

The visual Question Answer model should have the ability to understand both visual and linguistic capabilities. These models must possess some capabilities to function. They are locating an object (finding the location of any object within any given image mostly based on coordinate position along with height and width), finding object attributes (retrieval of attributes of any given object, e.g., color, shape, or any other traits of an object), activity being performed by an object (e.g., sport being played, running, walking, etc.), understanding of any given scene which is basically providing a high-level representation of the environment. The VQA models are prone to bias to remove this problem; multiple answers are normally being provided for any question.

This paper has been structured in the following way. Section 2 will give related existing work or literature study. This will be followed by Section 3, which will give a study of existing datasets available which is relevant for this analysis along with dataset details like licensing, year of formation, and other statistics about image, question, answer, etc. Section 4 will give details about the experiment being conducted along with an analysis of the results. Section 5 has conclusion details.

2 Background Study

Assessing the performance of the VQA model is a bit tricky. Regular ML models metrics like Accuracy (for classification problems), MAE (for regression problems), etc., cannot estimate performance properly. To effectively quantify the performance of a VQA model, the metric (Shrestha, R et al., 2017) must have the following capabilities:

a. Consistency: This depicts the ability to provide a consistent answer for a different form of the same question.

b. Grounding/Localization: The main purpose is better model interpretability. This is the ability to localize the region in the image which is relevant with respect to the answer.

c. Plausibility: Providing a justifiable answer, e.g., for a question like, is it a sunny day? The answer will be either yes, no, or cannot say something of this like.

VQA models generate an attention map to perform object localization with respect to the question being asked. The attention maps are nothing but a matrix that identifies a region within an image that is relevant for an image-question pair.A ranking score is generated based on a number of overlaps of bounding regions (Shalini Ghosh et al., 2019).
In a simple attention map mechanism mix of images and questions can derive whether a region is relevant or not with respect to that image. They propose a new architecture that represents the

image and question mix as a vector representation rather than pairwise confluence. This kind of vector representation can give accurate information about a region that is relevant for a specific context hence making this a context-aware region search (Xi, Y. et al., 2020). The authors proposed a very basic approach towards Visual Question Answer methods (Antol, S. et al., 2015). In this paper, the authors have shown the importance of captioning.

In a paper (Yang, Z. et al.,2016), the authors have adopted a stacked attention mechanism because the reasoning is complex; hence if multiple attention map mechanisms are stacked together, they perform well, as shown below how stacked attention map is able to pinpoint the region of an image which is related to a question. Figure 1 shows a sample example of how the stacked attention layer works for the question "What is sitting on the basket on a bicycle?"

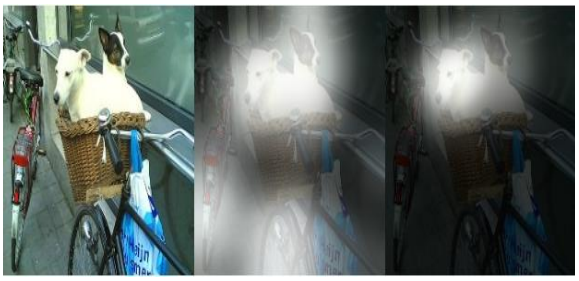


Figure 1: Multiple attention layers stacked together.

The stacked attention mechanism is a sequential approach, i.e., the next layer of the attention mechanism attempts to reduce the error caused by the previous attention layer. This approach is like a bagging or bootstrap aggregating algorithm. Another attempt is to try boosting approach, i.e., instead of sequential attention layer using parallel attention layer and at the result aggregates all attention layers data (like boosting concept). The use of multiple layers parallel manner and aggregating the results to get combined data works well and reduces the chance of overfitting.

In the paper (Goyal, Y. et al.,2017), the authors have emphasized on visual part or image part in VQA. The language part can be biased based on various human and other unavoidable biases.

In the paper (Anderson, P. et al., 2018), the authors proposed a mixed top-down and bottom-up approach for the generation of attention maps rather than the existing top-down approach for generating attention maps. The existing top-down approach is generating an attention map based on feature weights. The authors are claiming the attention maps are being generated for each feature. The proposed model generates the attentions map where each bounding box is associated with an attribute followed by an object, as shown in Figure 2.

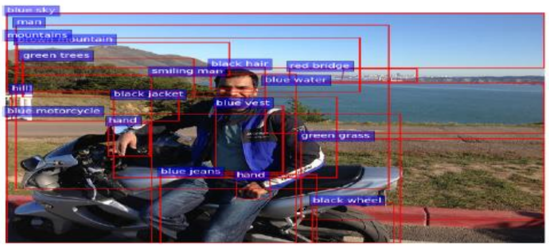


Figure 2: Attention map for a bottom-up approach.

In the above image, the top left bounding box is attached to "object: sky "and the attribute of sky object, "color: blue."

3 Related Datasets

This section is focused on the different dataset descriptions. Dataset is an important part of VQA.

DVQA dataset:

This dataset was developed in the year 2018. This is restricted only to research and educational purpose. This dataset also provides additional supplemental material. Along with the question-answer pair, detailed annotations of every object have been provided in the form of a bar chart. (Kafle, K. et al., 2018).

VQA v1 and v2 Dataset:

This dataset was developed in the year of 2016. This dataset has 256016 images. The images are inherited from the COCO image database. VQAv1 and VQAv2 have around 0.6M and 1.1M questions, respectively. The dataset has 7.9M answers. The dataset has 50K abstract scenes and 15K questions, and 1.9M answers to cater to the need of analysis abstract scenes (Zhang, P. et al., 2016).

VCR Dataset:

The dataset has 290K multiple choice questions along with their answers (290K). The dataset maintains answers are of 7.5 words on average. (Zellers, R. et al., 2019).

GQA Dataset:

This dataset was developed in the year 2019.The dataset has 110K images, where each image is associated with a scene graph of objects and relations. The dataset has 22M multistep questions.

A new metric also has been added to this dataset which is a combination of Accuracy, consistency, validity, and plausibility (Hudson, D. A. et al., 2019).

CLEVR:

This dataset was developed in the year 2018. The train set has 70K images and 700K questions, validation, and test sets have 15K images and 150K questions. Scene graph annotations are additional supplemental material available to boost the performance (Johnson, J. et al., 2018).

4 Experimental Result and Performance Analysis

For the experiment, we have used 30 epochs with minibatch gradient descent with a batch size of 128. Each epoch took almost 1.5 hours for the stacked area network model and roughly 2 hours for the VQA model.Now we will discuss internal model details.

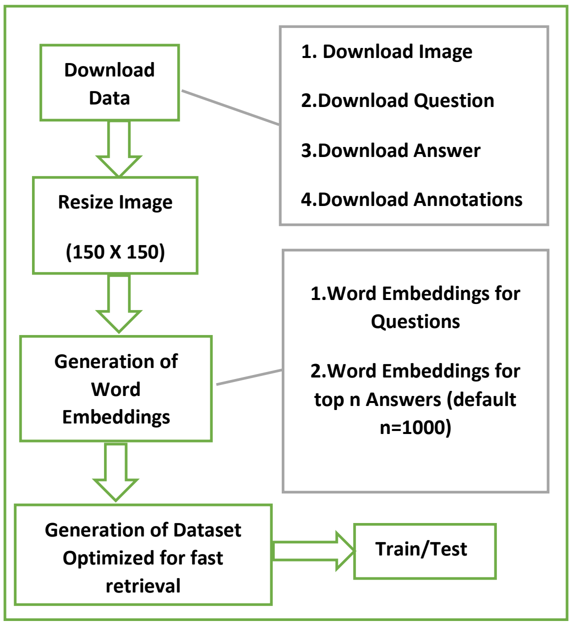

Download Data
1. Download Image
2.Download Question
3.Download Answer
4.Download Annotations
Resize Image
(150 X 150)
1.Word Embeddings for Questions
2.Word Embeddings for top n Answers (default n=1000)
Generation of Word Embeddings
Generation of Dataset Optimized for fast retrieval
Train/Test

Figure 3: Execution phases for VQA models.

Visual Question Answer (VQA) Model:

As stated earlier, in any VQA model, image and question-answer have been trained or processed separately, and at a later point in time, the feature vectors from image and text have been combined in a different manner. In the VQA model, we have used the pre-trained CNN model VGG19 to retrieve the image feature vector. For question-answer processing, we have used the Embedding layer and followed by a bidirectional LSTM layer to understand word embedding sequences pattern. During the experiment both image and text features are processed separately however we need to combine both the features so that the combined output can be fed into subsequent Neurons in Deep Learning frameworks. The image and text feature vectors have been combined using a torch tensor multiplication (like matrix multiplication). We have also ensured bypassing the argument to image and feature vector to have the same size. Followed by this step, we have used two hidden layers, and we have used tanh activation function in our hidden layer. The use of tanh activation function ensures we have bounded the output within -1 to 1 range. We have tried with relu and leaky relu but did not do well.

Multimodal Tucker Fusion for Visual Question Answering (MUTAN):

This model is similar to the VQA model with a small difference. In VQA, the image and text features are mixed using torch multiplication. However, in this model, image and text feature passed through separate deep neural network. Then the output from both the neural networks has been multiplied and forwarded to output generation.

Stacked Attention Network Model (SANM):

In this model also image and question-answer pairs have been processed in the same way as with the previous VQA model. However, there is a small change in combining image and text features. During the VQA model, we are doing simple torch tensor multiplication of text and image features, however, in this model, we are first creating a list of attention layers, and for each attention layer, we are combining image features and text features related to that attention map. Then all attention maps are stacked together, and they are further passed through a deep neural network with a single hidden layer where both image and text feature vectors have been passed with tanh as an activation function. Further, in the final layer, we have used softmax to extract which attention maps are relevant for that image-question pair. Then the combination of an image feature, word feature, and combined output from the stacked attention layer is passed through a shallow neural network for further processing.

Model Performance:

We have performed the experient on VQAv2 dataset. Table 1 shows the experimental results for models. As stated, earlier the VQA model performance is slightly better than the SANM model.

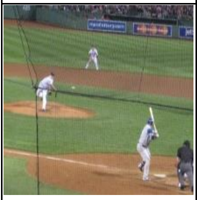
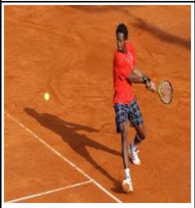
Image	Question	Output		
		SANM Model	VQA Model	Mutan
	Is the ball flying towards the batter?	'no' - 0.2597 'yes' - 0.2281 'both' - 0.1156	'yes' - 0.1507 'no' - 0.1015 'ground' - 0.0380 'floor' - 0.0192	'yes' - 0.2160 'no' - 0.1992 'night' - 0.1057 'red' - 0.0292
	What sport is being played?	'yes' - 0.2171 'no' - 0.1455 'old' - 0.0626 'both' - 0.0529 'new' - 0.0351	'tennis' - 0.1301 'baseball' - 0.0569 'unknown' - 0.0158 'none' - 0.0141	'yes' - 0.2466 'no' - 0.1710 'red' - 0.0520 'night' - 0.0375

Table 1: Experimental Results on test dataset

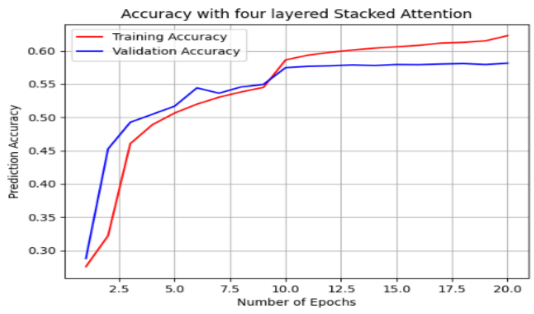


Figure 5: SANM model accuracy.

From Figure 5 we can see during epoch 8 or 9, the difference between training and validation loss was almost 0. We have taken that model as our best model since it reduces overfitting and generalizes well. The Accuracy is around 55%, and the loss is around 2.2.

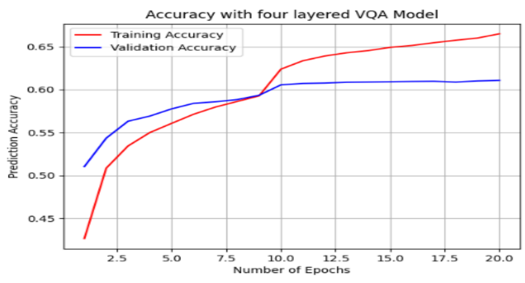


Figure 6: VQA model accuracy.

From Figure 6, we can see during epoch 8 or 9, the difference between training and validation loss is almost 0. We have taken that model as our best model since it reduces overfitting and generalizes well. The Accuracy is around 59%, and the loss is around 1.9. It is quite evident that the 2nd model is performing better than the 1st model.

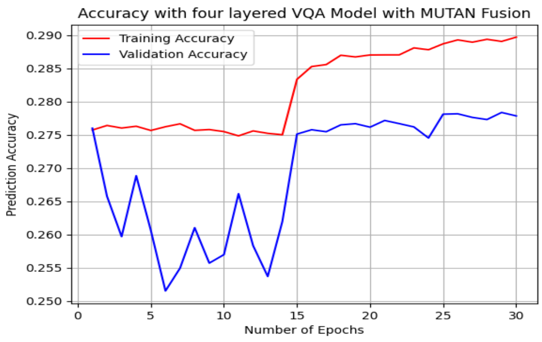


Figure 7: MUTAN model accuracy.

From Figure 7, we can see during epoch 14 or 15, the difference between training and validation loss is almost 0. We have taken that model as our best model since it reduces overfitting and generalizes well. The Accuracy is around 27%, and the loss is around 3.8. This model performance is not up to the mark. Overall VQA model is giving best accuracy among the 3 models evaluated.

5 Conclusions and Future Scope

Loosely speaking, so far, visual question answering models and datasets follow a specific pattern. Visual Question Answering dataset normally contains multiple images, which could be natural or synthetic, and along with this, each image contains a pair or combination of question and answer. Along with these in a few datasets, additional supplemental materials have been provided to support models as a feature to make better decisions. As per the VQA model is concerned with preprocessing image and text (question-answer) part preprocessing is going separately, i.e., no relation or dependency on each other. After preprocessing, the preprocessed data from image and text have been combined, which varies from model to model and then fed into Neural network models. This is overall architecture in simple terms. In this experiment, we have tested some of the VQA models successfully in our lab environment and produced results. We have also ensured to reduce overfitting by selecting the best model where training and validation loss difference is minimum.

References

Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., & Zhang, L. (2018). *Bottom-up and top-down attention for image captioning and visual question answering.* In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 6077-6086).

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., & Parikh, D. (2015). *Vqa: Visual question answering.* In Proceedings of the IEEE international conference on computer vision (pp. 2425-2433).

Cadene, R., Ben-Younes, H., Cord, M., and Thome, N. (2019). *Murel: Multimodal relational reasoning for visual question answering*. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 1989-1998).

Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., and Parikh, D. (2017). *Making the v in vqa matter: Elevating the role of image understanding in visual question answering.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 6904-6913).

Hedi Ben-younes, Rémi Cadene, Matthieu Cord, Nicolas Thome. (2017). *MUTAN: Multimodal Tucker Fusion for Visual Question Answering.* In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV).

Hudson, D. A., and Manning, C. D. (2019). *GQA: A new dataset for real-world visual reasoning and compositional question answering*. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 6700-6709).

Johnson, J., Hariharan, B., Van Der Maaten, L., Fei-Fei, L., Lawrence Zitnick, C., & Girshick, R. (2017). *CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning.* In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2901-2910).

Kafle, K., Price, B., Cohen, S., and Kanan, C. (2018). Dvqa: *Understanding data visualizations via question answering*. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5648-5656).

Kafle, K., & Kanan, C. (2017). *An analysis of visual question answering algorithms.* In Proceedings of the IEEE International Conference on Computer Vision (pp. 1965-1973).

Shalini Ghosh, Giedrius Burachas, Arijit Ray, Avi Ziskind: *Generating Natural Language Explanations for Visual Question Answering using Scene Graphs and Visual Attention*: arXiv:1902.05715v1, 2019.

Shrestha, R., Kafle, K., and Kanan, C. (2019). *Answer them all! Toward universal visual question answering models*. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 10472-10481).

Singh, A., Natarajan, V., Shah, M., Jiang, Y., Chen, X., Batra, D., and Rohrbach, M. (2019). *Towards vqa models that can read.* In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 8317-8326).

Suhr, A., Lewis, M., Yeh, J., and Artzi, Y. (2017, July). *A corpus of natural language for visual reasoning.* In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (pp. 217-223).

Xi, Y., Zhang, Y., Ding, S., & Wan, S. (2020). *Visual question answering model based on visual relationship detection.* Signal Processing: Image Communication, 80, 115648.

Yang, Z., He, X., Gao, J., Deng, L., and Smola, A. (2016). *Stacked attention networks for image question answering.* In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 21-29).

Yi, K., Wu, J., Gan, C., Torralba, A., Kohli, P., and Tenenbaum, J. B. (2018). *Neural-symbolic vqa: Disentangling reasoning from vision and language understanding*. arXiv preprint arXiv:1810.02338.

Zellers, R., Bisk, Y., Farhadi, A., and Choi, Y. (2019). *From recognition to cognition: Visual commonsense reasoning*. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 6720-6731).

Zhang, P., Goyal, Y., Summers-Stay, D., Batra, D., and Parikh, D. (2016). Yin and yang: *Balancing and answering binary visual questions*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 5014-5022).

Deep Embedding of Conversation Segments

Abir Chakraborty
Microsoft India
abir.chakraborty@gmail.com

Anirban Majumder
Flipkart Internet Private Ltd.
majumder.a@flipkart.com

Abstract

We introduce a novel conversation embedding by extending Bidirectional Encoder Representations from Transformers (BERT) framework. Specifically, information related to "turn" and "role" that are unique to conversations are augmented to the word tokens and the next sentence prediction task predicts a segment of a conversation possibly spanning across multiple roles and turns. It is observed that the addition of role and turn substantially increases the next sentence prediction accuracy. Conversation embeddings obtained in this fashion are applied to (a) conversation clustering, (b) conversation classification and (c) as a context for automated conversation generation on new datasets (unseen by the pre-training model).

We found that clustering accuracy is greatly improved if embeddings are used as features as opposed to conventional tf-idf based features that do not take role or turn information into account. On classification task, a fine-tuned model on conversation embedding achieves accuracy comparable to an optimized linear SVM model on tf-idf based features. Finally, we present a way of capturing variable length context in sequence-to-sequence models by utilizing this conversation embedding and show that BLEU score improves over a vanilla sequence to sequence model without context.

1 Introduction

Embedding of natural language units (word, sentence or paragraph) deals with the problem of finding a vector space representation of these units that can be used in downstream applications of classification, summarization or token identification. For example word embeddings (Mikolov et al., 2013a,b,c; Pennington et al., 2014) have found application in information retrieval (Manning et al., 2008), document classification (Sebastiani, 2002; Kim), question answering (Tellex et al., 2003; Minaee and Liu, 2017), named entity recognition (Turian et al., 2010) and parsing (Socher et al., 2013). Extending the same concept to sentences and documents one can also find the corresponding vector representations independently (Le and Mikolov, 2014) or by suitably averaging the word vectors (Kusner et al., 2015).

While aforementioned embeddings are created without optimizing for (or even considering) downstream applications there are recent approaches that seek optimal representations based on pre-training (Radford et al., 2018; Howard and Ruder, 2018; Peters et al., 2018). These applications can be at sentence-level such as natural language inference (Bowman et al., 2015; Williams et al., 2018) and paraphrasing (Dolan and Brockett, 2005) where the semantic relationship between sentences are captured or at word-level tasks (Rajpurkar et al., 2016; Wang et al., 2018). There are two different approaches for applying pre-trained embeddings, (a) feature-based ((Peters et al., 2018), where the model architecture is task-specific and pre-training is a feature of the architecture) and (b) fine-tuning (Radford et al., 2018; Devlin et al., 2019) (where the pre-training architecture is quite generic to handle a variety of downstream tasks and model parameters are later fine-tuned for specific tasks). While most of the pre-training architectures use unidirectional language models (Radford et al., 2018; Peters et al., 2018), Bidirectional Encoder Representations from Transformers (BERT, (Devlin et al., 2019)) uses a different strategy to learn sentence/paragraph representations and achieves best scores on a variety of tasks.

Even though there are different strategies for creating word, sentence and document level embeddings, there is no study available in literature that deals with *conversation embedding*. While a piece of conversation may look very similar to a paragraph (and one can probably start with paragraph embedding to embed conversations) it has two important additional pieces of information, namely, turns and roles. A turn can consist of a single word, sentence or multiple sentences (all belong to a single role) and a conversation can have many participants (roles) where it is crucial to distinguish who is saying what. An efficient representation of a complete conversation (or part of it) should take into account the role and turn information (and their congruence) for downstream applications.

The most important application of conversation embedding is in the area of automated dialogue generation. Starting with the vanilla sequence-to-sequence model (Sutskever et al., 2014; Vinyals and Le, 2015) there are different approaches to capture the "context" so that meaningful responses can be generated (Sordoni et al., 2015b; Mei et al., 2017). The "context" continuously grows as the conversation progresses and can be defined in terms of everything that has happened in the conversation so far or key words from earlier turns extracted by some attention based algorithms (Bahdanau et al., 2015). There could be different approaches to capture a context, e.g., (a) separate RNNs for previous turns and roles, (b) attention over previous turns or (c) a global vector representing counts of tokens from previous turns etc. However, all of them have limitations either in capturing all the required information or in their ability to deal with a continuously increasing context length. An embedding that can map a variable length context (i.e., a conversation segment) into a numeric vector while including key pieces of information required for generating the next response would be immensely helpful in automatic dialogue generation. This is what has been attempted in this work where we create conversation embedding using BERT and apply to various downstream tasks. Our contributions are

1. Extension of BERT based *sentence representation* to *conversation representation* by adding the notion of roles and turns and thus creating an embedding of conversation segments hitherto unavailable in literature.

2. We show that with the inclusion of roles and turns during pre-training the next sentence prediction accuracy increases.

3. Application of these pre-trained models on conversation clustering shows better accuracy over tf-idf based features.

4. We demonstrate how conversation embedding can be used to capture context in sequence-to-sequence models and thereby improving the BLEU score.

2 Related Work

Very little work is available in the literature on conversation embedding, especially that treats conversations with all its associated complexities. Most of the work has been on word embedding (non-neural, (Brown et al., 1992; Ando and Zhang, 2005; John Blitzer and Pereira, 2006; Pennington et al., 2014) and neural (Mikolov et al., 2013a,b,c; Liu et al., 2017)), sentence embedding (Le and Mikolov, 2014) and embedding of paragraphs (Dai et al., 2015). Recent approaches involving pre-training and fine-tuning also deal with sentences and sentence pairs (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2019) and the down-stream tasks are mostly classifications (and not sequence generation). On conversation embedding, the closest that we see is from Mehri et al. (2019) where multiple pretraining objectives are explored. Conversations are encoded using recurrent neural network (RNN) and no information from roles or turns are included.

The importance of capturing "context" for relevant response generation is well understood. (Sordoni et al., 2015b) tried capturing the context initially using bag-of-words representation (Sordoni et al., 2015a) and later by a hierarchical recurrent encoder-decoder (HRED) approach (Serban et al., 2016a) applied to the movie dataset (Banchs, 2012) with only one previous utterance appearing as the context. Here, a dialogue D consisting of a series of utterances $\{U_1, U_2, \ldots, U_M\}$ was decomposed as

$$p_\theta(U_1, U_2, \ldots, U_M) = \prod_{m=1}^{M} \prod_{n=1}^{N_m} p_\theta(w_{m,n} | w_{m,<n}, U_{<m}) \quad (1)$$

and two encoders were used to encode context and current input. A second approach based on stochastic latent variables (called VHRED model) to capture dependency amongst multiple time steps is proposed by (Serban et al., 2016b) and it was shown that VHRED generated responses preferred over HRED. HRED architecture is further modified by (Li et al., 2018) to take bidirectional GRU as input to the encoder, for application to multi-modal input scenario (Agarwal et al., 2018) and in a GAN set up for generating a sequence response (Su et al., 2018).

In the next section we describe our approach of creating conversation embedding from BERT. Subsequently, we demonstrate three applications of this embedding, namely, clustering, classifications and dialogue generation. Finally, conclusions are drawn.

3 Conversation Embedding Using BERT

BERT represents a sentence or a pair of sentences by their Wordpiece tokens (Sennrich et al., 2016), segment names (A or B) and token-wise positions. While this representation is enough for contiguous sentences from a paragraph it does not take into account (1) role or speaker and (2) turn or depth of the conversation. In this work we add these two additional pieces of information along with the rest of the embedding. To give an example, Table 1 is a snippet of a typical conversation with the corresponding roles (Customer and Agent) and turns $(0, 1, 2, \ldots)$:

The corresponding text, role and turn tokens for input to a BERT model will be as shown in Fig. 1. While the role in the present case takes only two values, $i.e.$, agent and customer (although there is no restriction) the turn can be as high as 200 in our conversation data. Thus, the turn embedding can be similar to the position embedding used in Transformer, $i.e.$, sines and cosines. However, in the present work we have projected both the turn and role values to a fixed-dimensional vector and learnt the corresponding embeddings (Gehring et al., 2017). These embeddings are added together along with the position and segmentation embeddings defined in the original BERT paper.

The pre-training steps of BERT are partly based on tasks defined earlier, namely, masked language model (MLM) and next sentence prediction (NSP). However, in case of MLM the masked word can be from different roles and turns. Similarly, the sentence pairs in NSP can span across multiple roles and turns. Thus, both MLM and NSP will drive better understanding of the conversation structure.

Task #3: Middle Sentence Prediction

In addition to NSP and MLM we have also introduced a new task, namely, middle sentence prediction (MSP). As the name suggests, we choose any two alternate turns (both coming from the same role) and try to predict the middle turn (different role). Similar to the NSP task, MSP is also converted into a binary classification problem where 50% of the time the actual middle turn is chosen and for the rest of the time another turn from a different conversation is picked randomly (while maintaining the role). In this way, the model should be able to understand the conversation structure better that will also help in applications like automated response generation.

4 Experiments

The data for all the experiments presented here are from conversations between customer service agents and existing/new customers who contact the customer service for their order related issues. The conversations between customer and agents are divided into sessions which we merged together to generate a single conversation. For BERT model pre-training we have randomly selected 100,000 chats of various different topics. These conversations are of varying number of turns and tokens (as can be seen in Table 2). For text normalization following steps are carried out (a) lower casing, (b) replacing entities like number, customer, city and state names, url, date, ticket number etc. by their corresponding tokens and finally (c) removing empty spaces, multiple punctuation and special characters. We have restricted the number of words in the vocabulary to 30,000 (same as what was considered in the original BERT paper (Devlin et al., 2019)).

4.1 Pre-training

We use the base configuration (Devlin et al., 2019) for pre-training, $i.e.$, number of layers, $L = 12$, hidden dimension, $H = 768$ and number of self-attention heads, $A = 12$. Both turn and role are projected into the same hidden dimension H as used for segments and positions. There are only two words in the role vocabulary (”A” and ”C”) whereas we have taken 128 as the turn vocabulary dimension. The maximum sequence length used in all the examples is 128.

We have created two different pre-training datasets from the 100,000 chats. The first one does not contain any MSP task and has 1.7 million examples for NSP and MLM tasks created by having a duplication factor of 5 (for random masking and

Turn	Role	Text
0	**Agent**	Please proceed with your query.
1	**Customer**	1want my order delay for one day. On the date of 26 nov
2	**Agent**	I certainly understand your concern. Let me check that.
3	**Customer**	Okk
4	**Agent**	Thanks for waiting. On checking details ...

Table 1: Sample conversation and the corresponding text, roles and turns

Statistics	Turn	Token
Minimum	1	1
Maximum	195	8409
Mean	13	23
50^{th} percentile	11	14
75^{th} percentile	17	26
90^{th} percentile	25	52

Table 2: Statistics of Turns and Tokens in the pre-training dataset

binary label selection). The second dataset has all possible ($\sim$1M) MSP examples (no duplication) along with NSP and MLM examples created with a duplication factor of 2 (678k) resulting in 1.77 million data points. Thus, the two datasets are similar in size but having a totally different distribution of task types. We use a batch size of 32 for all examples and train for 3 epochs ($\sim$170k steps), which takes around 80 hours in a 12 GB Tesla GPU machine.

The corresponding MLM and NSP task performance for these datasets are shown in Table 3. The effect of adding the role and turn is clear in the next sentence prediction with a much better accuracy. It can also be seen that adding MSP task increases (next or masked) sentence prediction accuracy. Addition of role and turn on the other hand has little effect on the MLM accuracy, mostly because these masked words are more dependent on surrounding words (rather than the roles or turns of the surrounding words).

Once we have pre-trained BERT models (with roles and turns) we can use the representation of [CLS] (from the top layer or from multiple layers, (Devlin et al., 2019)) as a representation (embedding) of the entire conversation. This representation (a vector of length H, 768 in the present case) then can be used for many potential downstream predictions as it has captured the entire conversation in a fixed length vector. Here we apply our pre-trained BERT models for clustering and automated response generation. For all the applications, including MSP, we use the representation of the [CLS] token at the top layer as an embedding. In addition, we also fine tune the BERT model for intent prediction.

4.2 Conversation Clustering

The data for conversation clustering consist of a different set of 50,000 conversations (again taken from conversations between customers and agents). Each conversation has a labeled intent (based on agents' tagging of the corresponding issue) and the distribution of these intents in the dataset is shown in Table 4. Each conversation is converted into a fixed length feature vector (dimension 768) using the pre-trained models described in the previous section. A t-SNE (van der Maaten and Hinton, 2008) plot of this dataset is shown in Fig. 2 where interactions amongst different intents and existence of multiple intents in a conversation are captured to a certain extent. For example, most of the intents have some overlap with "others" intent and "status" (order) is closely related to "delivery". Also, conversations with "others" tag and falling in the range of component-1 > 0 and component-2 > 10 seem to have no overlap with any of the other existing intents. A sample of 10 conversations from this region shown below confirms that:

- check my last order i want to know about which battery inbuilt
- can u update the name from [[name]] [[name]] to [[name]] [[name]] in the invoice ?

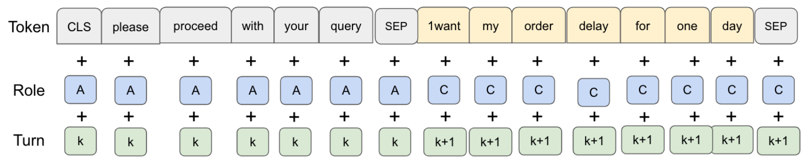


Figure 1: Conversation segments and corresponding BERT input representation using tokens, roles (A - agents, C - customer) and turns (k is the turn number varying between 1 and #turns).

Type	MLM Accuracy	NSP Accuracy
Set-1, no role and turn	82.4%	89.0%
Set-1, with role and turn	83.5%	96.4%
Set-2 (MSP), with role and turn	81.6%	97.5%

Table 3: Pre-training test results on the two different datasets

Intent	Percentage
Cancel	11.2
Delivery	16.5
Return & Refund	18.3
Status Inquiry	22.1
Others	31.9

Table 4: Distribution of pre-defined intents in the clustering dataset

- [[order-id]] item missing product
- i do not want to change the address i want to change the payment method
- what is mean by no cost emi ?
- hi . i want to remove my default mobile num [[phone]] and change to [[phone]] i have the option to change email but not mobile
- i want to buy screen guard with phone but u charge extra delivery charges for screen guard yes
- call me back
- what is the offer for this phone i am not able to understand hello can you type you there
- ji mujhe emi pe phone lena h

where the last sentence has both English and Hindi words.

Although tags are not always very accurate and there can be multiple intents in a single conversation, we apply k-means algorithm (with 5 clusters) on the BERT embedding and calculate the accuracy. We also extract tf-idf based features (uni-grams and bi-grams) from the conversations (without taking role and turn information into account) and apply

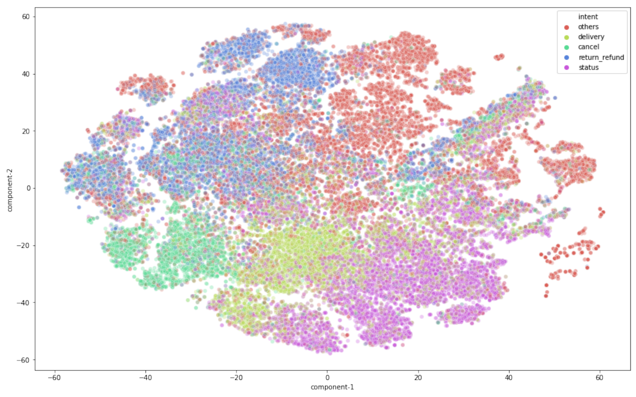


Figure 2: Clustering of conversations by t-SNE (van der Maaten and Hinton, 2008) applied on conversation embeddings

Feature	Accuracy
tf-idf based feature	31.07%
word2vec based feature	33.28%
BERT embedding (no MSP)	44.08%
BERT embedding (with MSP)	42.98%

Table 5: Accuracy of k-means clustering of conversations

k-means algorithm with 5 clusters. Once cluster numbers are obtained for individual data points we apply Hungarian algorithm (Papadimitriou and Steiglitz, 1998) to map cluster number to intents (class names) and compute the accuracy. Instead of clustering all 50000 data points we randomly sample 10000 data points 10 times and apply k-means algorithm 5 times (for both tf-idf and BERT based feature representation) and compute the average accuracy. The results are shown in Table 5 where it is clear that BERT based feature (on an unseen dataset) has resulted in a much better accuracy underscoring the efficiency of this approach of embedding a conversation.

4.3 Conversation Classification

For classification we consider another set of 40,000 conversations (different from what is used in pre-training or clustering) with intents (labels) provided by the customer service agents. The distribution of these intents in this data is similar to what is shown in Table 4. We fine-tune the pre-trained BERT model for 32,000 data points and apply on the rest 8,000 examples. We use the same vocabulary of pre-training (with 30,000 words) for tokenization with maximum sequence length of 128. We fine tune the BERT model for 5 epochs with a batch size of 32 and learning rate of 2×10^{-5}. For comparison, we consider a linear SVM model with tf-idf based features. Hyper-parameters and ranges shown in Table 6 are considered for grid search. The results are presented in Table 7. It can be seen that BERT fine-tuned model achieves comparable accuracy (slightly higher) on completely unseen data.

hyperparameter	range
maximum document frequency	0.5, 0.75, 1.0
n-gram range	1, 2, 3
idf usage	True, False
tf-idf norm	L1, L2
α	10^{-4}, 10^{-5}, 10^{-6}
Regularization	L2, elasticnet

Table 6: Hyperparameter set for SVM classifier

Model	Accuracy
Linear SVM	72.04%
BERT fine-tuned	72.13%
BERT fine-tuned (with MSP)	72.24%

Table 7: Accuracy of different classifiers

4.4 Conversation Generation

The final example shows another application of BERT based features for automatic dialogue generation. As discussed previously, sequence-to-sequence (or $seq2seq$) models are not naturally amenable to accommodate conversation contexts and various approaches have been tried in the past. We try to generate response tokens r_t^k for turn k by maximizing the probability of response $\mathbf{r}^k = \{r_1^k, r_2^k, \dots, r_T^k\}$

$$p(\mathbf{r}^k|\mathbf{i}^k) = \prod_{t=1}^{T} p(r_t^k|r_1^k, \dots, r_{T-1}^k, \mathbf{i}^k, \mathbf{c}^{1:k-1}) \tag{2}$$

where $\mathbf{i}^k$ and $\mathbf{c}^{1:k-1}$ are the input and context for the kth turn, respectively. Since the context contains everything that has happened so far in the conversation, $i.e.$, $\mathbf{c}^{1:k-1} = \{\mathbf{i}^1, \mathbf{r}^1, \mathbf{i}^2, \mathbf{r}^2, \dots, \mathbf{i}^{k-1}, \mathbf{r}^{k-1}\}$ it is an ever growing list and difficult to encode in a fixed length vector. In this work, we convert a variable length context into a fixed length feature vector using a pre-trained BERT model, i.e.,

$$\mathbf{c}^{1:k-1} = BERT(\{\mathbf{i}^1, \mathbf{r}^1, \mathbf{i}^2, \mathbf{r}^2, \dots, \mathbf{i}^{k-1}, \mathbf{r}^{k-1}\}) \in \mathbb{R}^H \tag{3}$$

where H is the embedding dimension in BERT model.

Figure 3 shows the schema of applying context embedding. The model is based on an encoder-decoder pair modified for context embedding. At k-th turn, the context embedding represents turns 1 to $k-1$ that is used as an initial hidden state to the encoder (after a linear transformation). Next, tokens of the k-th turn are fed into the encoder one-by-one and the corresponding encoder outputs are recorded. The final encoder output (h_5 in Fig. 3) is concatenated with the context embedding (again with another linear transformation) and used as the initial state of the decoder. Following Bahdanau style attention (Bahdanau et al., 2015) the decoder state is compared with encoder outputs to compute attentions weights that are applied on the encoder outputs to get 'context vector'. This context vector is concatenated subsequently with the $k+1$-th turn tokens before given as input to the decoder to generate decoder outputs. Although not used in this work, context embedding can also be included in the attention weight calculation.

The data for conversation modeling is also taken from prior customer interaction with agents. However, we have considered only conversations for a specific issue, $i.e.$ 'status check'. We have manually extracted 3,872 conversations with 14,978 turns (data points) that have only this intent and no other additional intents displayed in the same conversation. The median number of turns in these conversations is 6 (75^{th} percentile is 8 and 90^{th} percentile is 15). We fixed the encoder sequence length to 32 and decoder sequence length to 128 (90^{th} percentile is 128). The context (which is a cumulative of previous turns) that is passed to the BERT model ideally should be less than 128 (the maximum sequence length considered in the BERT model). However, in our dataset the maximum number of tokens in a context is 363 while 99^{th}

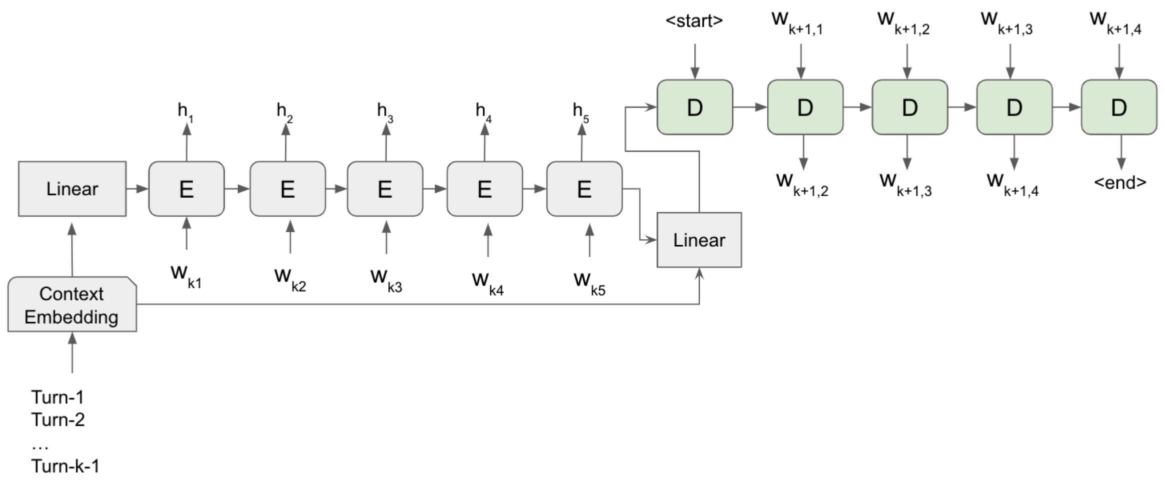


Figure 3: Encoder-decoder model with context embedding. Context embedding is used to set the initial hidden state of both the encoder (E) and decoder (D).

percentile is 109. Thus, for less than 1% of the cases the context will be truncated before being evaluated by BERT model.

We have used GRU for encoder and decoder (single layer) with a hidden dimension of 256 (same for the token embedding dimension). The inputs are reversed before given as input to the encoder. Before passing the context embedding (vector of dimension 768) to the encoder and decoder linear layers with ReLU activation (projecting 768 to 256) are used. In case of decoder there is an additional linear layer (with ReLU activation) that acts on the concatenated vector of last encoder hidden state and context vector (i.e., of dimension $768 + 256 = 1024$).

We have used 80% of the data for training and 20% for validation. All the model components are trained by Adam optimizer (Kingma and Ba, 2014) with default values and batch size of 16. Model performance is evaluated by BLEU score (Papineni et al., 2002) where the validation data is evaluated at the end of every epoch and tested for improving BLEU score. If the score does not improve for 4 consecutive epochs training is terminated. Table 8 shows the BLEU scores (BLEU-2 and BLEU-3 indicate BLEU scores for bi-grams and tri-grams) with and without conversation embeddings. The best rating is obtained when MSP task was not included in BERT pre-training which is not intuitive. However, both conversation embedding based models result in a better BLEU score than vanilla seq2seq model.

Model	BLEU-2	BLEU-3
no context embedding	0.2284	0.2037
with MSP	0.2354	0.2116
without MSP	**0.2403**	**0.2177**

Table 8: BLEU-2 and BLEU-3 for conversation response generated by different seq2seq models

5 Conclusion

We have introduced an embedding (representation) of a conversation (or conversation segment) by augmenting role and turn information to word tokens and utilizing BERT for pre-training. This pre-trained model can be used either to generate features from new conversations or can be fine-tuned further on specific tasks. In this work we have explored both the options. Pre-trained model based conversation features are used for (a) conversation clustering and (b) for representing contexts in a conversation for predicting the next response. In case of clustering we show that embedding based features result in higher accuracy when compared to tf-idf based features. Similarly, for conversation modeling embedding feature based context representation drove higher BLEU score when compared to a vanilla seq2seq model without any contextual information. We also fine-tune a pre-trained model for conversation classification on new dataset and obtain accuracy similar to what is given my a linear SVM model trained on tf-idf based features. With these examples we show the general applicability of the current approach on modeling various tasks involving conversation data.

References

Shubham Agarwal, Ondrej Dusek, Ioannis Konstas, and Verena Rieser. 2018. Improving context modelling in multimodal dialogue generation. In *INLG*.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Rafael E. Banchs. 2012. Movie-dic: a movie dialogue corpus for research and development. In *ACL*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. *EMNLP*.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Andrew M. Dai, Christopher Olah, and Quoc V. Le. 2015. Document embedding with paragraph vectors. In *NIPS Deep Learning Workshop*.

Jacob Devlin, Ming-Wei, Chang Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. *Proceedings of the Third International Workshop on Paraphrasing*.

J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. 2017. Convolutional sequence to sequence learning.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *ACL*.

Ryan McDonald John Blitzer and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. *Proceedings of the 2006 conference on empirical methods in natural language processing*, page 120–128.

Yoon Kim. Convolutional Neural Networks for Sentence Classification.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization.

Matt J. Kusner, Yu Sun, Nicholas I. Kolki, and Kilian Q. Weinberger. 2015. From Word Embeddings To Document Distances. *ICML*.

Quoc V. Le and Tomas Mikolov. 2014. "Distributed Representations of Sentences and Documents. *ICML*, pages 1188–1196.

Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Christopher Joseph Pal. 2018. Towards deep conversational recommendations. *ArXiv*, abs/1812.07617.

Li-Ping Liu, Francisco J. R. Ruiz, Susan Athey, and David M. Blei. 2017. Context selection for embedding models. In *NIPS*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Sch. 2008. *Introduction to Information Retrieval*. Cambridge.

Shikib Mehri, Evgeniia Razumovskaia, Tiancheng Zhao, and Maxine Eskenazi. 2019. Pretraining methods for dialog context representation learning.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2017. Coherent dialogue with attention-based language models. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. *ICLR Workshop*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *NIPS*, pages 3111–3119.

Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. *HLT-NAACL*.

Shervin Minaee and Zhu Liu. 2017. Automatic Question-Answering Using A Deep Similarity Neural Network.

Christos H Papadimitriou and Kenneth Steiglitz. 1998. Combinatorial optimization: algorithms and complexity. *Courier Corporation*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. *EMNLP*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *xxx*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, OpenAI.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *EMNLP*.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM Computing Surveys*, 34:1–47.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. *ArXiv*, abs/1508.07909.

Iulian Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016a. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*.

Iulian Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2016b. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. Parsing With Compositional Vector Grammars. *ACL*, pages 1–47.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jing Nie. 2015a. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. *ArXiv*, abs/1507.02221.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jing Nie, Jianfeng Gao, and William B. Dolan. 2015b. A neural network approach to context-sensitive generation of conversational responses. In *HLT-NAACL*.

Hui Xu Su, Xiaoyu Shen, Pengwei Hu, Wenjie Li, and Yun Chen. 2018. Dialogue generation with gan. In *AAAI*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, pages 3104–3112.

Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes, and Gregory Marton. 2003. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the SIGIR Conference on Research and Development in Informaion Retrieval*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the ACL*, pages 384–394.

Oriol Vinyals and Quoc V Le. 2015. A neural conversational model. *Proceedings of the 31st International Conference on Machine Learning*, pages 3104–3112.

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. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. *NAACL*.

DialogActs based Search and Retrieval for Response Generation in Conversation Systems

Nidhi Arora
Interactions, LLC
narora@interactions.com

Rashmi Prasad
Interactions, LLC
rprasad@interactions.com

Srinivas Bangalore
Interactions, LLC
sbangalore@interactions.com

Abstract

Designing robust conversation systems with great customer experience requires a team of design experts to think of all probable ways a customer can interact with the system and then author responses for each use case individually. The responses are authored from scratch for each new client and application even though similar responses have been created in the past. This happens largely because the responses are encoded using domain specific set of intents and entities. In this paper, we present preliminary work to define a dialog act schema to merge and map responses from different domains and applications using a consistent domain-independent representation. These representations are stored and maintained using an Elasticsearch system to facilitate generation of responses through a search and retrieval process. We experimented generating different surface realizations for a response given a desired information state of the dialog.

1 Introduction

A good conversation system is the one that enables its users to converse freely in natural language text. To handle conversations in a robust manner, the system should have set of responses covering many possible ways the end-customer can interact with the system. Response generation is a challenging problem for spoken dialog systems, with the quality of the generator depending on factors such as adequacy, fluency, variation and readability (Stent et al., 2005). Partly because of the need to adapt to requirements of the specific domain for which the system is designed, many deployed applications, including the ones that provide context for our work, follow a template-based approach, in which response templates with possible slot fillers are manually authored by application designers. Such responses largely satisfy the quality measures of adequacy and fluency, where adequacy measures whether the response conveys the intended meaning completely, non-redundantly, and unambiguously, while fluency measures linguistic correctness and appropriateness of style. However, there can be challenges with respect to variation and readability.

Variation is intended to avoid repetitiveness so that the responses in multi-turn dialogs sound natural and human-like, while readability ensures that responses are interpretable in their dialog context. For example, asking users for basic personal information or task specific details is a common task across many business needs. However, repeatedly using the same small set of scripts such as `What is your account number?` or `Please share your account number` has the undesirable effect of sounding predictable and unnatural.

With a team of experts authoring prompts for diverse applications across different domains, collecting response variations for different response types for various stages of a conversation is effortless. The challenge however is that the collection is useful only if the variants are maintained with dialog state specific equivalence classes that are consistent across domains for authors to access and reuse. Fig 1 presents some alternative realizations for frequently occurring use cases of asking customers for their name. While $r1$ is simply querying the end user to provide their name, $r2$ acknowledges the capture of information and requests for confirmation of correctness. Responses $r3$ - $r5$ are used when the customer has not provided the desired information during the prior turn and needs to be prompted again; these are paraphrases providing reasons for why the information is necessary.

The motivation behind this work is to cluster these variations using a systematic and consistent

r1 Please say and spell your first name for me.
r2 I heard your first name as NAME, is that correct?
r3 I need your name in order to continue.
r4 Can I have your name to start the order.
r5 I'm sorry, but I need your name to book your appointment

Figure 1: Different variations of asking the end-user customer about their name.

framework to promote response sharing across different domains and to suggest possible variations to choose from. In this paper, we describe our preliminary work on dialog act classification of a large database of responses authored by a design team for different applications using a consistent and common domain independent annotation scheme. These response variants can then be used by the designers to provide possible alternative realizations of the content they want to convey. Furthermore, since selecting an appropriate variant is partly a function of the dialog context, our future goal is also to develop a context aware response suggestion model that can account for readability through the proper use of context-dependent elements such as referring expressions (e.g., *Please share your account number with me* vs. *Please share that with me*) and discourse markers (e.g., *What is your account number?* vs. *And what is your account number?*).

Our main contributions in this paper are:

- Creating a unified taxonomy for the communicative function of dialog acts that is universally applicable across different domains and covers probable agent/user tasks.

- Using Elasticsearch to maintain a repository of system responses and transforming context independent response generation problem into a search and retrieval task.

Section 2 presents our approach to Dialog Act (DA) classification, focusing on the taxonomy for the communication function (CF) component. Section 3 explains the use of Elasticsearch for maintaining a repository of utterances indexed along many different dimenions. Section 4 presents our preliminary experiments on classification and static response generation.

2 Communicative Function of Dialog Acts

The meaning of an utterance in dialog has long been characterized as a dialog act, designed to capture the communicative behavior of a participant in terms of speech acts (Austin, 1962; Searle, 1969). Many DA schemes have been developed over the years, but most were designed for specific domains and applications (Allen et al., 1994; Allen and Core, 1997a; Anderson et al., 1991; Alexandersson et al., 1997). Here, we adapt the ISO standard for DA annotation (Bunt et al., 2012) because it provides a domain independent representation that covers a broad range of intents for all aspects of a conversation state.

Based on the information-state update approach to meaning in dialogue (Bunt, 2000; Traum and Larsson, 2003), a DA in the ISO framework (ISO-DA) has two main components: a semantic content (SC), which describes the entities, events, actions, properties or relations that the DA is about, and a communicative function (CF), which specifies how addressees should update their information state with the semantic content. For example, the utterance *What is your account number?* has some representation of the customer's account number as the semantic content, while the CF should represent the fact that the value of this entity is not known and that it is being requested of the customer.

Since the focus here is on the CF classification of responses, we have adapted the ISO-DA CFs to reflect the commonly occurring functions in our data of approximately 37K unique response utterances. The following provides the CFs, with definitions and examples. Broadly, the functions can be classified as the ISO-DA categories of information-seeking functions, information-providing functions, commissives and directives. For some of the CFs, such as *query_info_intro*, further refinements are needed, however, our preliminary goal for this work is to explore the feasibility of the classification task with a coarse-grained taxonomy. Additionally, of the nine dimensions in the ISO-DA taxonomy (Bunt, 2006), we have focused here on classifying CFs in the task-related dimension. CFs that fall in other dimensions such as turn management, feedback, and social obligations are all treated as an *other* category but will be handled in future work.

query_info: Information-seeking function where the customer is asked to provide the unknown value of a specific entity, such as the request to provide the value of the check-out date in "*what date will you be checking out?*" or of the birthdate in "*please say or enter the 2-digt month, 2-digit day and 4-digit year of your birth.*"

query_info_open: Open-ended information-seeking function where the customer is asked to provide their intent, e.g., "*How may I help you?*" This includes requests for the intent related to a specific topic, such as "*what would you like to know about call blocking?*"

query_info_intro: In some dialog contexts, explicit requests for information are preceded by a statement that some information is needed, such as when a prior explicit elicitation for information was not successful for some reason, and an explanation is provided for the specific request E.g., "*I need your routing number in order to process your payment*". Because these responses are not explicit requests, we believe that their function is of the information-providing type rather than the information-seeking type.

query_confirm: Information-seeking function with an explicit request to confirm (or disconfirm) a proposition, such as "*I heard your credit card number as $NUM. Is this correct?*" or "*Just to be sure, I am about to cancel your annual subscription service. Is that correct?*" The expected response from the customer in such cases is a "yes" or "no".

query_disambig_yn: Information-seeking function to elicit a "yes" or "no" response from the customer, but unlike *query_confirm*, this does not elicit a confirmation. Utterances with this function are typically used to suggest an action to the customer to move the task in some direction in the dialog flow, for example, to invite the customer to transfer to a live agent for some task, as in "*Would you like to talk to someone about renewing?*", or to accept help via email, as in *Would you like me to send you an email to help you reset your password?*"

query_disambig_select: Information-seeking function to present choices for selection, such as "*Is this for a business, an educational institution or for a government entity?*" or "*Would you like to pay this with a debit card, credit card, or a different payment method?*" In the current version of the taxonomy, this does not distinguish between selection between entities and selection between actions.

promise: Commissive function committing to perform some action, such as "*I'll send a link to the email we have on file for you so you can reset your password.*"

offer: Commissive function also committing to perform some action, but unlike *promise*, the commitment here is contingent on some condition which may or not be specified. E.g., "*I can get you help with your login.*", "*I can get you to someone who will help with gift cards, but I just need bit more detail.*"

deflect_request: This is a special case of a commissive function that occurs frequently in our data, and involves deflecting a request from the customer while suggesting an alternative course of action. Typically, the deflected request is for a live agent, with examples such as "*I understand you want to speak to someone, but ...*"

instruct: Directive function specifying some action that the customer should undertake, such as "*Enter your username and password and click 'sign in'.*"

inform_issue: This is a special case of an information providing function to inform the customer of some contrary to expectation situation, such as when the customer's utterance in the prior turn was not understood or captured, e.g., "*I wasn't able to hear what you just said.*"

inform: This covers a broad class of information-providing utterances. Examples include "*Your confirmation number is $NUM.*", "*I see you have a tax appointment on $DATE at $TIME.*"

other: This category was used for utterances that could not be captured by any of the other CFs. As mentioned above, these include utterances with CFs from non task-related dimensions. We observed that most of these involve feedback CFs and social obligation CFs.

The CF taxonomy was developed first over a seed set of 200 utterances and validated over successive iterations as part of the active learning experiments described in Section 5. For example, the utterance "*I understand you want to speak to someone, but if you give me your credit card number, I can process your payment for you.*" has three functional segments with three CFs: *deflect_request*, *query_info_intro* and *offer*. In this stage of our work,

Field	Description
$uspan$	complete response as collected from our applications
cf	communicative function as annotated in Section 2.
$entities$	entities of interest such as bank account
$vertical$	domain names for ex. finance, tourism, hospitality etc.
$uspan\_vector$	dense vector representation for the utterance

Table 1: Description of indexing fields.

we ignore the ordering of the segments, and therefore, the classes as well.

3 Response Generation

For designing robust conversational systems, including ours, there exists a team of experts who list down alternative ways of how end customer might interact with the system and create responses for each such use case individually. The wording of the prompt has to be carefully chosen both to convey the desired message and to query for further information. This process is repeated from scratch for each new client and application, even though similar prompts may have been authored in the past for similar scenarios. For example, asking users for their personal information for authentication is a common task across many different applications. We observed many such situations where very similar system responses were present in different applications but were created from scratch because of no means available to access and re-use responses generated in the past.

In this section we describe the mechanism we devised for maintaining a repository of responses that can be used either for designing new conversation flows or for reuse directly as templates. As mentioned above each system response is characterized by both the communicative function and semantic content. We thought of providing search interface where designers could mention one or more of these dimensions to specify these requirements and access different realizations of the response they want to generate. One dimension is to provide a text span describing the theme of the current response such as validating gift cards, informing about longer wait times, calling about a new product launch, querying for name or date of birth. Communicative functions provides another dimension to search and filter responses by the desired intent. While we need an exact match to search for communicative functions, text based specifications should be able to retrieve responses that are semantically similar to the specified constraints.

Our requirements prompted us to use Elasticsearch because it facilitates both exact search as required for CFs as well as similarity based search in case designer describes a text phrase to specify key aspects of the content they wish to communicate. ElasticSearch (ES) is an open source search and analytics platform widely used for non-structured text data, hence it perfectly matches our requirements. Table 1 provides the list of fields indexed in ES for our task along with their definitions.

Each indexed field helps to filter responses according to the desired specifications, for example that the entity must be $billing\_address$ or $account\_number$. These filters help to obtain responses that are appropriate for a given context. The communicative function would be $query\_info$ when asking for $phone\_number$ or $first\_name$ of a person, however, *Please say and spell your first name* is more appropriate than *Please say and spell your phone number.* Since the current system implementation is context dependent, search fields such as $entities$ help to provide some context specific information thereby retrieving responses that are more relevant to the current context.

4 Models

In order to investigate response generation, we need to annotate a collection of system responses with the set of communicative functions defined above. The annotation task at hand essentially is a classification task, where given a system prompt we want to predict the relevant communicative function. For example, given system response " *Can you please tell me the phone number associated with this account*", output should be *query_info* and given input utterance as "*I can help with automatic payments, but first for security purposes please share the phone number linked to this account* " model should predict three labels as *offer, query_info_intro.*

Many machine learning classifiers are available in the literature for supervised multi-class classification problem such as SVM, KNN, and Gradient boosting etc, but being supervised algorithms they

cf	No of Prompts
Collections	1200
Communications	4000
Financial Services	9000
Food services	1909
Hospitality	5278
Insurance	6846
Retail Services	6266
Utilities	2160

Table 2: Distribution of

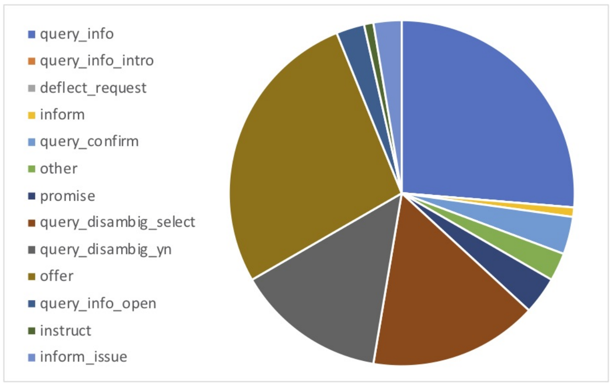


Figure 2: Different variations of asking the end-user customer about their name.

require annotated dataset for the models to learn patterns and make predictions whereas we had no reference dataset available with us. The only reference training dataset available with ISO annotation scheme is DialogBank (Bunt et al., 2016) which is very small to be used for training such classifiers and also more generic than ours.

We collected a set of approximately 37K system responses from twenty different applications across eight different domains providing enough response variations for commonly occurring modules. These responses are a subset of around 2 billion system responses being used by our conversation assistants on a daily basis for various clients across different domains. The distribution of different domains is present in Table 2.

BERT (Devlin et al., 2019) and T5 (Raffel et al., 2019) are two of the most widely adopted transfer learning approaches that are known to yield reasonably good performance even for a very small data set. For this study, we used Huggingface implementation of BertForSequenceClassification, where the final hidden state of the sequence is input to a fully connected softmax layer with cross-entropy loss function. There are two standard approaches to train multi-label classifier. The first being to train individual binary classifiers for each class in one-vs-rest approach and the second is to list down all possible combinations treating each one as an independent class. For example, if there are 3 unique classes, a, b and c, then we can have at most 7 distinct class labels (a), (b), (c), (a, b), (a, c), (b, c), (a, b, c) where each combination is treated as an atomic class. The latter approach though works well for smaller set of unique labels but becomes difficult as the number of classes increase and the distribution varies a lot.

Of the two approaches for solving multi-label classification problems, the preliminary set of experiments indicated that training ensemble of individual binary classifiers resulted in better performance than treating each combination as an atomic class. This implies that we trained and saved individual classifiers for each of the 13 communicative functions as binary classifiers. We experimented with different learning rates and found best results for 1e-5 with batch size 16 and max epochs 20. Also, to account for varied distribution of labels, we computed class weights for each class label as $size(label\_i)/max$, where max is over all sizes.

5 Experiments

We conducted experiments in two phases. The first phase is to train multi-label classifier for classifying system responses into space of communicative functions with a reasonable degree of accuracy. In the second phase, we analysed the quality of responses retrieved by the system for both simple queries given only CF or textual description and complex queries defined using a combination of other querying dimensions.

5.1 Multi-label Classification

We begin the training process by manually annotating a subset of about 200 responses. The training set had responses with the number of CFs varying from 1 to 3, with the ratios 0.60, 0.30 and 0.10 respectively. We then adopted an active learning approach to train and validate batches from the un-annotated corpus and adding them to the labelled data set. The distribution of communicative function over first 200 samples is presented in Fig. 2. The initial distribution clearly indicates that more than 50% of the prompts were information seeking prompts with labels "*query_info*" and "query_disambig". Also note that the initial

cf	**F1-score**
$deflect\_request$	1.0
$query\_info$	0.875
$query\_confirm$	0.833
$query\_info\_open$	1.0
$query\_info\_intro$	0.875
$query\_disambig\_select$	1.0
$query\_disambig\_yn$	0.698
$inform$	0.95
$inform\_issue$	1.0
$instruct$	1.0
$promise$	0.91
$offer$	0.96
$report$	0.5
$other$	0.615

Table 3: F1-scores for communicative functions at the end of training process.

distribution had no training utterances for "deflect_request", "query_info_intro" and some other relatively occuring combinations such as "*offer, query_info*.

These classifiers were used to predict next batch of 200 prompts that were manually verified by the authors and language experts from the design team. With each iteration of training, predictions and evaluation, the classification accuracy improved finally leading to state-of-the-art performance score of 85% at the end of our training process. Though not directly comparable with joint intent and slot prediction models (Qin et al., 2020)(He et al., 2021) due to difference in the training objective, we observed that the accuracy scores are in comparable range of current literary works. We repeated our training-prediction-manual evaluation cycle four times increasing the test set from 100 to 500 samples. Each time, the predictions were manually verified with a team of designers discussing and agreeing to the final set of CF label. We adopted manual verification process because the communicative intent can not always be explicitly inferred from the wording of the response.

Table 3 reports the F1-scores at the end of round four with a training dataset of 1100 system responses. We found the F1-scores reaching to their maximum performance scores for most of our class labels and felt that the current classification model can be used to annotate our repository with reasonable accuracy. One of the reason for lower accuracy but higher F1-score was that certain percentage of responses were not predicted any CF label. Overall this percentage was close to 1%, where none of the classifiers were confident to assign the CF label. On inspection, we found that this was for those cases where either the system response has been incomplete or the wording of the prompt was such that the classifier could not predict any class label with a higher degree of confidence. Also, for the response labels $query\_confirm$ and $query\_disambig\_yn$, the responses were difficult to annotate clearly even by human experts. We are extending our training dataset with more such examples and hope to increase the accuracy level while keeping F1-scores at the maximum.

Once we could annotate the collection of system responses with a reasonable degree of accuracy, we created a repository that can be used by the design team to retrieve prompts by either specifying the communicative functions or by providing an abstract description of the current dialog state.

5.2 Response Generation using ES

It is an unrealistic assumption for the design team to learn a new technology (Elasticsearch) and a new language of communicative functions to specifying the desired information state. We therefore created a GUI based user interface for designers. to enter their requirements. The interface internally converts the search filters and their values into the query language executed by the Elasticsearch. We experimented with different search filters (query combinations) and found that given sufficient information, the system generated response variations consistent with the specifications mentioned by the designer.

Generating responses for a specific communicative function simply transitioned to executing a boolean query over the Elasticsearch. Table 4 provides subset of sample responses generated for the criteria ($cf = inform$). From the perspective of Elasticsearch, the results were 100% accurate but from the perspective of using these prompts for the current context we found them not directly usable. As no other semantic information was available about the dialog state, the responses retrieved by the system are coming form various domains and dialog state level.

From Table4 we can observe that there is one response informing customer about payments made, another response mentions mailing address while another is for street address and so on. In the ab-

response
Thank you, I have successfully submitted your payment.
I see that you have a repair issue that is scheduled to be resolved on DATE.
The mailing address we have on file for you is WORD.

Table 4: Variations for communicative function:$inform$

response
I see your street address is.
I have your street address as.
The street address I have on file for you is.

Table 5: A subset of 3 response variations generated for prompt:"street address" and communicative function:"inform"

sence of context specific information, the design choice is left to the designers to select which variation is more appropriate for the current context. As searching only by CFs would lead to data abundance problem, there are two different ways to specify context specific information; by selecting entities or by providing an abstract description of the content. One such example is presented in Table 5 where the designer filters the responses by including street address in search criteria. As we can see, almost all the responses are semantically similar to each other and can be adapted by the designers for the current conversation state.

As another example, Table 6 presents the scenario where the user provides a text based description and does not specify any communicative function explicitly. The system returns three different kinds of responses that look very similar but have different communicative intents behind each. The first response informs the customer about longer wait times and offers to help fulfill the desired task. The second response on the other hand provides the reason for longer wait times whereas the third response only informs the customer about the current situation. By providing three different variations, the system can reveal how these cases have been previously handled and provides an option to reuse any one of these realizations as per the current context.

Using a combination of both communicative function and text description provides the most appropriate means to specify the search requirements. We tested 25 different queries specifying both the text specification of the content and the context appropriate communicative function and observed the quality of system responses returned. We used Mean Reciprocal Rank to evaluate the set of responses generated given only text based specifications. We executed 30 different queries using a mixture of simple text based descriptions and complex queries with both components textual description and communicative functions. We found average MRR scores of 0.6, 0.71 and 0.72 for Top-1, Top-3 and Top-5 respectively with Universal Sentence Embedding (USE) for computing semantic similarity. The MRR scores for ELMO and SBERT were much lower for our datasets.

6 Literature Review

Accurately predicting the speakers communicative intent is extremely important for a successful communication and thus intent detection has always been widely pursued research thread. As virtual assistants are becoming a part of daily life, it has been acknowledged that most a times speaker is communicating multiple aspects with in a single utterance. There is an increasing trend towards training joint models for intent detection as Multi-Label Classification (MLC) and entity detection(also called slot filling (Hou et al., 2021) (Qin et al., 2020) (He et al., 2021). These systems compute relevance score for each label and utterance combination and then select the labels with maximum similarity score. Some of these approaches are few shot learning approaches proposing techniques to perform MLC with fewer examples, but they all pretrain on domain specific data and then extend this to out-domain dataset. In contrast, our work aims to annotate with domain-independent dialog act labels and only focuses on predicting communicative functions, hence we adopted conventional machine learning approaches for classifying communicative functions.

The concept of representing dialog acts using domain independent general purpose schemas has been studied multiple times as Dialog Act Markup in Several Layers (DAMSL) by Allen et. al (Allen and Core, 1997b) and as ISO standard by Bunt et.al. (Bunt et al., 2012). The ISO taxonomy pro-

cf	response
$offer$	Due to heavy call volume at this time it could take over 90 minutes to talk to a representative, lets see if I can help you.
$inform\_issue$	We are sorry for your inconvenience, however, we are experiencing extremely high call volume due to the recall and this has caused extremely long wait times to connect with an agent
$inform$	Wait a moment while this call is being transferred to our system.Wait times are longer due to heavy call volumes.

Table 6: Variations of the system response informing customer that there are excessive wait times.

vided generic representations of a speakers intent by defining 9 core dimensions and around 60 different communicative functions using domain independent and task independent labels.

7 Conclusion

In this paper, we proposed a taxonomy of communicative functions that effectively captures the communicative intent of a dialog turn using domain independent labels providing means for flexible and generic dialog modelling. The taxonomy was used to annotate a subset of user responses from human-machine conversations used by our real-life applications on day-to-day basis. We experimented with this annotated dataset to generate different linguistic variations of the system responses for given communicative function and desired keywords indicating the essence of the current dialog turn. Our experiments indicated that the proposed taxonomy can successfully learn representations that capture what the dialog is written to accomplish across different applications and verticals. We experimented with these annotations in a dialog generation settings and found that we are able to generate system responses given desired specifications from the existing data itself.

References

Jan Alexandersson, Bianka Buschbeck-Wolf, Tsutomu Fujinami, Elisabeth Maier, Norbert Reithinger, Birte Schmitz, and Melanie Siegel. 1997. Dialogue acts in VERBMOBIL-2. Technical Report 226.

James Allen and Mark Core. 1997a. DAMSL: Dialog act markup in several layers (Draft 2.1). Technical report, University of Rochester, Rochester, N.Y.

James Allen and Mark Core. 1997b. Draft of DAMSL: Dialog act markup in several layers. Unpublished manuscript.

James F Allen, Lenhart K Schubert, George Ferguson, Peter Heeman, Chung Hee Hwang, Tsuneaki Kato, Marc Light, Nathaniel G Martin, Bradford W Miller, Massimo Poesio, and David Traum. 1994. The TRAINS project: A case study in building a conversational planning agent. Technical Report 532, Computer Science Department, University of Rochester, Rochester, N.Y.

Anne H Anderson, Miles Bader, Ellen Gurman Bard, Elizabeth Boyle, Gwyneth Doherty, Simon Garrod, Stephen Isard, Jacqueline Kowtko, Jan McAllister, Jim Miller, Catherine Sotillo, Henry Thompson, and Regina Weinert. 1991. The HCRC map task corpus. *Language and speech*, 34(4):351–366.

John L. Austin. 1962. *How to do things with words*. Cambridge: Oxford University Press.

Harry Bunt. 2000. Dialogue pragmatics and context specification. *Abduction, Belief and Context in Dialogue. Studies in Computational Pragmatics. Amsterdam: Benjamins*, pages 81–150.

Harry Bunt. 2006. Dimensions in dialogue act annotation. In *LREC*, pages 919–924.

Harry Bunt, Jan Alexandersson, Jae-Woong Choe, Alex Chengyu Fang, Koiti Hasida, Volha Petukhova, Andrei Popescu-Belis, and David Traum. 2012. ISO 24617-2: A semantically-based standard for dialogue annotation. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 430–437.

Harry Bunt, Volha Petukhova, Andrei Malchanau, Kars Wijnhoven, and Alex Chengyu Fang. 2016. The dialogbank. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016*. European Language Resources Association (ELRA).

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.

Ting He, Xiaohong Xu, Yating Wu, Huazhen Wang, and Jian Chen. 2021. Multitask learning with knowledge base for joint intent detection and slot filling. *Applied Sciences*, 11(11).

Yutai Hou, Yongkui Lai, Yushan Wu, Wanxiang Che, and Ting Liu. 2021. Few-shot learning for multi-label intent detection. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 13036–13044. AAAI Press.

Libo Qin, Xiao Xu, Wanxiang Che, and Ting Liu. 2020. TD-GIN: token-level dynamic graph-interactive network for joint multiple intent detection and slot filling. *CoRR*, abs/2004.10087.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

John R. Searle. 1969. *Speech acts: An essay in the philosophy of language*, volume 626. Cambridge university press.

Amanda Stent, Matthew Marge, and Mohit Singhai. 2005. Evaluating evaluation methods for generation in the presence of variation. In *Proceedings of CICLing*.

David R Traum and Staffan Larsson. 2003. The information state approach to dialogue management. In *Current and new directions in discourse and dialogue*, pages 325–353. Springer.

An On-device Deep-Learning Approach for Attribute Extraction from Heterogeneous Unstructured Text

Mahesh Gorijala **Aniruddha Bala** **Pinaki Bhaskar** **Krishnaditya Vikram Mupparthi**
Advanced Technology Lab (ATL)
Samsung R&D Institute India - Bangalore (SRI-B)
{m.gorijala, aniruddha.b, pinaki.b, krish.aditya, vikram.m}
@samsung.com

Abstract

Mobile devices, with their rapidly growing usage, have turned into rich sources of user information, holding critical insights for betterment of user experience and personalization. Creating, receiving and storing important information in the form of unstructured text has become a part and parcel of daily routine of users. From purchase deliveries in Short Message Service (SMS) or Notifications, to event booking details in Calendar applications, mobile devices serve as a portal for understanding user interests, behaviours and activities through information extraction. In this paper, we address the challenge of on-device extraction of user information from unstructured data in natural language from heterogeneous sources like messages, notification, calendar etc. The issue of privacy concern is effectively eliminated by the on-device nature of the proposed solution. Our proposed solution consists of 3 components – A Naïve-Bayes based classifier for domain identification, a Dual Character and Word based Bidirectional Long Short Term Memory (Bi-LSTM) and Conditional Random Field (CRF) model for attribute extraction and a rule-based Entity Linker. Our solution achieved a 93.29% F1 score on five domains (shopping, travel, event, service and personal). Since on-device deployment has memory and latency constraints, we ensure minimal model size and optimal inference latency. To demonstrate the efficacy of our approach, we have experimented on CoNLL-2003 dataset and achieved comparable performance to existing benchmark results.

1 Introduction

With an estimated 3.5 billion active users or about 80% of all mobile subscribers, Short Message Service (SMS) was the most widely used communication application in the past few years [1]. Even with the advent of social media and messenger applications, communication and information storage in digitised form are vastly prevalent via SMS, notifications, calendar invites and mail. Some examples we readily see are casual conversations with a friend over SMS, online shopping related notifications and event booking details, just to name a few.

According to 2020 Annual report by CTIA [2] , there were 2.1 trillion text messages exchanged worldwide, an increase of 52 billion messages since 2019. According to SMS marketing statistics for 2020/2021 reported by FinancesOnline [3] , 98% of SMS are opened compared to only 20% of emails and 95% of the read SMS are responded to within 3 minutes of delivery. Moreover, SMS is still the most powerful marketing tool for businesses with 75% of customers preferring receiving offers via SMS. The CTR for text messages is much higher (9.18%), compared to other marketing channels such as Google Adwords (1.91%) and Facebook (0.90%).

Apart from SMS and notifications, researchers have investigated other potential data sources like calendar, email, user utterances and communication logs for extracting information. In fact, many establishments are making use of the personal knowledge extracted from different sources on smartphones to provide better service. For example, Google extracts and summarizes travel, event and accommodation reservation information from emails [4] . However, most of the published literature is focussed on singular sources of information and/or certain domains of interest like bio-medical,

[1]https://en.wikipedia.org/wiki/SMS

[2]www.ctia.org/news/report-2020-annual-survey-highlight

[3]https://financesonline.com/sms-marketing-statistics/

[4]https://developers.google.com/gmail/markup/reference/#reservations

Data Source	Information Extracted	Inference Drawn
Messages	Shopping, travel and financial activities, event attendance, service availed etc.	Preferred vendors, products, venues, payment modes etc.
Calendar	past and upcoming events and occasions	Preferred relations
Call & Message logs	caller, callee, message sender, receiver details	Frequent caller, callee, message sender and receiver
Notifications	All the above details and activities	User's details, preference and interest

Table 1: Particular data items extracted and high-level inferences drawn from the data sources.

events, etc.

In this paper, we address the challenge of on-device extraction of user information from unstructured data in natural language from heterogeneous sources, which include SMS, notifications, calendar etc. Our proposed system offers a unique method for efficient on-device functionalities. We achieve 93.29% F1 score on five domains (shopping, travel, event, service and personal). The system is implemented as a service on the device. The information that is obtained from these data sources with a preliminary analysis and the high-level inferences sought from it, are summarized in Table 1. The abundance of personal information on smartphones can hence be safely utilized for many apps like recommender systems, virtual personal assistants, on-device content presentation, to provide better services to end users. This provides a holistic view about the user encompassing users' behaviours, interests, activities, etc.

A key consideration is the constant ongoing conflict between the service provider's desire to track the consumer and the consumer's concern for the privacy of their data. This issue is effectively addressed by the on-device nature of the proposed system, letting the user enjoy its benefits conveniently as the data processing is limited to local environment.

Some of the features afforded by this new dimension of user's data, which enables personalized device intelligence, are as follows:

- User's attention can be proactively drawn to offers and discounts regarding the products of only the categories they wish to purchase, filtering all the annoying spam.
- Enable simplified interaction with smart assistant
- Event reminders can be triggered appropriately
- Convenient grouping or reordering of SMS/ Notification/ Calendar data according to user preferences
- Assist in better planning of activities, for instance, booking airport cab with prior knowledge of user's travel plans
- Recommender services based on understanding of user's shopping behaviour or preferred types of events (like concerts, sports matches, photography, art etc.)

2 Related Work

Digital communication devices continue to offer a growing variety of personalized services to enhance user experience. This is facilitated by increased access and extraction of user information available in both structured and unstructured forms. Structured data, generally consisting of text entered in template fashion or in any pre-defined format (like date, zip code etc.), can be conveniently processed whereas unstructured text (like Short Message Service (SMS), Notifications, Calendar events etc.) poses multiple interesting challenges.

Firstly, apart from emanating from heterogeneous sources on the device, unstructured data on mobile devices does not always conform to grammatical correctness, rendering it difficult for most of the existing Natural Language Processing (NLP) techniques better suited for formal grammar. Secondly, most of the advanced information extraction techniques demand server-based deployment, raising privacy concerns of user data storage on cloud. There is limited exploration on on-device information extraction from unstructured text, befitting its memory and latency constraint requirements. Thirdly, owing to the special nature of data in consideration, there is a lack of standardised benchmark datasets. Most of the previous works have shown a significant amount of research effort being directed to collection, curation and pre-processing of short-text corpus. After data procurement comes the daunting task of annotation based on the identified guidelines of entities and attributes relevant to each domain of interest.

There is some pre-existing work on domain classification and information extraction from email, SMS, notifications and social media text. Most of the previous works handling SMS are primarily focused on spam-filtering or certain rudimentary levels of classification. Almeida et al. (2011) and Cormack et al. (2007), limit the task to binary classification of SMS as spam or non-spam. Dewi et al. (2017), explore the possibility of multi-class classification of messages into 4 categories with limited data instances. They achieve best results with logistic regression. In comparison, we categorize messages into 6 classes and perform further information extraction.

With regard to information extraction from short texts like SMS and notifications, traditionally various approaches have been investigated including use of POS taggers, regular expressions, hidden Markov models (HMM), logistic regression, specific syntactic parsers or a combination of the above. Jiang et al. (2010) investigate the extraction of named entities related to events or activities from Chinese SMSes in handsets, using Hidden Markov models (HMM). Although their method achieves a lower F-score on a small SMS corpus of 1,000 messages, the authors significantly reduce the memory consumption. Polifroni et al. (2010) implement logistic regression to recognize name, date, location and time entities from messages. Their reported F-scores for names and locations reaches 88 on an individual word basis, but they do not report on computational or memory resources required of their approach or exact corpus size. Cooper et al. (2005), exploit the syntactic structure in messages and used pattern matching for extraction. Since pattern matching is not robust to variations in data, Ek et al. (2011) complement pattern matching with a logistic regression based classifier.

Recent works on SMS and notifications involves the use of deep learning models for information extraction. Vatsal et al. (2020) implement a hybrid hierarchical LSTM-CNN architecture for SMS classification and then use class specific entity parsers based on pattern matching. Li et al. (2018) use the insight that notifications are formatted using templates. Templates are extracted using longest common subsequence mining and then clustered using DBSCAN algorithm. Template semantic rules are then generated using a Bi-LSTM network.

We believe ours is the first work that provides a generalized deep learning architecture for information extraction from multiple unstructured data sources. We also categorize inputs into multiple domains and link the attributes to pre-existing entities in the database. Our system pipeline has been designed to cater to multiple applications such as customization services, recommender systems, knowledge base population, etc. requiring the holistic understanding of users.

3 Proposed Methodology

The proposed information extraction pipeline consists of 3 major components – Domain Classifier, Attribute Extractor and Entity Linker. A pictorial representation of the pipeline is depicted in Fig. 1.

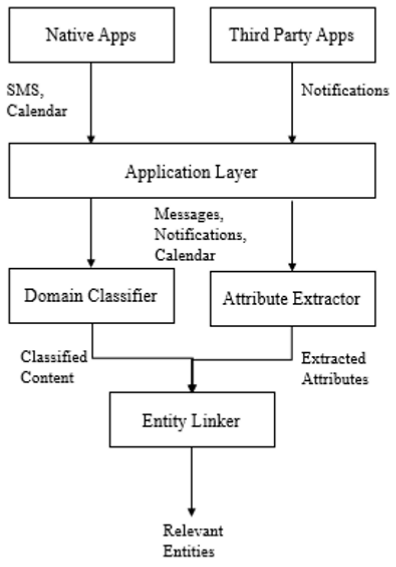


Figure 1: System Overview

3.1 Data Preprocessing

We converted messages and notifications into a more generalised format by using pattern matching. All date and time variations were mapped to ⟨DATE⟩ and ⟨TIME⟩ tokens respectively and currency values were mapped to ⟨CURRENCY⟩ tokens. All other numeric values were replaced by ⟨NUM⟩ and alphanumeric values were converted to

⟨ALPHANUM⟩ token. Website URLs were identified and replaced by ⟨URL⟩ token. For e.g. the input sentence

> dispatched : your package with philips dj shl3000 / 00 over - ear headphone (blue) will be delivered on or before wed , june 29 . track at www.amzn.in/track

will be processed as

> dispatched : your package with philips dj ⟨ALPHANUM⟩ / ⟨NUM⟩ over - ear headphone (blue) will be delivered on or before ⟨DATE⟩ , ⟨DATE⟩ . track at ⟨URL⟩

3.2 Domain Classifier

This module classifies the unstructured part of the data into one of a set of predefined domains. For this work, the predefined set includes Shopping, Travel, Event, Service, Personal & Spam. This classification allows the identification of user's domain-wise preferences, which increases the accuracy of the recommendations / ranking generated by applications using extracted information. Since the domains are very distinct, we propose a hybrid Naïve Bayes model for this simple text classification task. The block diagram for proposed approach is shown in Fig. 2. We augmented the standard Naïve Bayes model with an n-gram language model to effectively capture the inherent template-like structures in the data. The probabilities were computed based on tf-idf of tokens along with its length and the sender (in the case of messages and notifications).

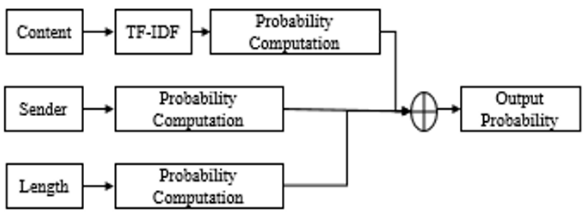


Figure 2: Domain Classifier Architecture

The domain classifier is used in the downstream task of triplet generation. For e.g. for an SMS where a token is tagged as Start Date or End Date, we will use the domain classifier output to decide the property name before adding the triple to the knowledge graph. For instance, if the domain of the SMS is shopping then we would know that the tagged date corresponds to product delivery date. We also tried to use the domain information as an input to our attribute extraction model, however, it did not improve the overall performance.

3.3 Attribute Extractor

This module extracts a predefined set of attributes (listed in Table 3) from the unstructured part of the data, which contain the pieces of information about the event / activity being conveyed by the data. Attribute extraction is modelled as a sequence labelling task. We implement a dual character and word embedding based Bi-LSTM (Hochreiter and Schmidhuber, 1997) followed by a CRF (Lafferty et al., 2001) trained on the sequence labelled set of messages, notifications and calendar. In the training dataset, all attribute tokens in a training sample are appropriately marked with one of the a) "B" for beginning, b) "M" for middle and c) "E" for end token, followed by the attribute type tags. Non-attribute tokens are marked with "O" (other) tag. E.g. "Your order for Samsung Galaxy S20 will be delivered today" is marked as "O O O B-Product M-Product E-Product B-Status M-Status E-Status B-EndDate". All tokens in the training dataset with frequency greater than 5 are included in vocabulary and the remaining tokens are substituted by the ⟨UNK⟩ token. We also create a character vocabulary required for generating character based word embeddings. Using the two vocabularies, we generate word and character lookup tables that are required for tokenization of input.

Fig. 3 describes the process of generating embeddings for each token in the input sentence. The input sequence is first encoded using the word lookup table and then passed into an embedding layer, which gives us W_{emb}. Each character is then considered as a token and encoded using the character lookup table. The output is passed into bidirectional $LSTM_{char}$ which generates forward and backward representations. These are then concatenated to give character based word embedding. Ultimately, the word embedding and the character based word embedding are concatenated to give the final token embedding. The computations performed inside LSTM cells are as follows:

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \tag{1}$$

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f) \tag{2}$$

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o) \tag{3}$$

$$g_t = tanh(w_g[h_{t-1}, x_t] + b_g) \tag{4}$$

$$c_t = f_t * c_{t-1} + i_t * g_t \tag{5}$$

$$h_t = o_t * tanh(c_t) \tag{6}$$

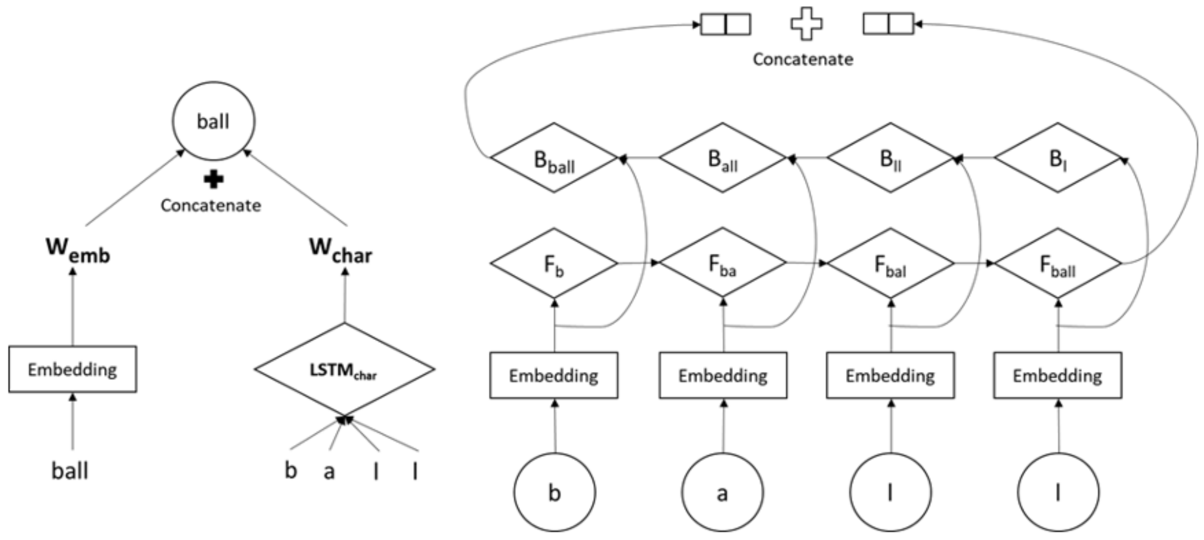


Figure 3: Word Embedding Generation in Dual Bi-LSTM: The representation of a word is formed by concatenating the word embedding and the character level representation of the word from char LSTM.

Where i_t, f_t, o_t and g_t are input, forget, output and cell gates respectively, x_t is input at time step t, h_t is hidden state and c_t is cell state. The hidden states at final time step are considered as representation of input sequence. The proposed attribute extractor model is depicted in Fig. 4. The input token embeddings are fed into the Bi-LSTM encoder followed by a fully connected layer. This generates emission scores, which represent likelihood of word being a certain tag. The role of the CRF layer is to model the joint likelihood of the entire tag sequence. This is achieved by calculating the transition scores, which represent the likelihood of word being a certain tag given the previous word was a certain tag. For decoding, Viterbi algorithm is used to find the tag sequence with maximum likelihood.

3.4 Entity Linker

This module links the entities identified by the attribute extraction module to an appropriate entity in the existing database having either the same or different name. It also processes the "Sender" information (for messages and notifications) and "Date" information (for calendar events) and accordingly adds entities if they weren't identified by the attribute extractor from the main content. As entity linking module gets diverse attributes as its input, it's implemented with different approaches.

- We use off-the-shelf string matching algorithms such as FuzzyWuzzy [5] based on Levenshtein Distance and phonetic algorithms such as Soundex [6] for linking the vendor attribute.
- We use an ontology and a predefined set of rules to match Source Location, Destination, Travel Mode, Travel Class, Event Type, Service Type, Status, Relationship and Occasion attributes.
- In case of Start Date, Start Time, End Date and End Time, we identify all possible variations in our data and map them to a standard format using pattern matching.
- Some attributes like ID, Product, Vehicle Number, Event Name and Amount are left unmatched.

4 Dataset

For data collection we sent out an organization wide broadcast seeking voluntary participation from users with diverse demographics . For this purpose, the users were required to install an application developed by the team. The application masked the user's private information such as name, contact number, financial details etc. and allowed the user the option to filter messages before sharing.

We collected $\sim$ $90K$ $(90,811)$ messages, $\sim$ $55K$ (54990) notification and $\sim$ $1K$ calendar

[5] https://pypi.org/project/fuzzywuzzy/
[6] https://pypi.org/project/soundex/

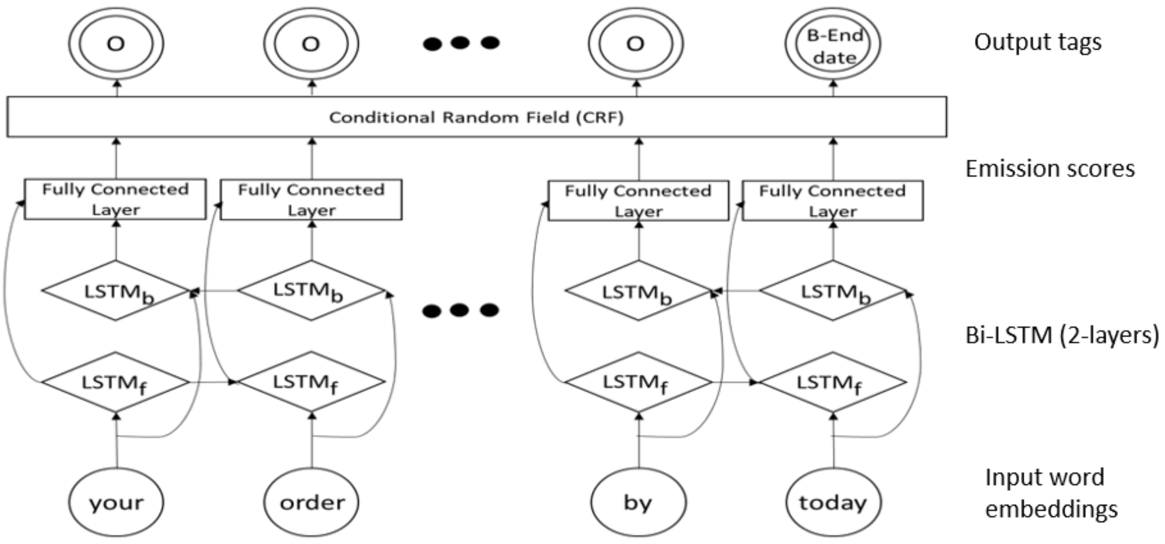


Figure 4: Attribute Extractor Architecture

Dataset	Domains	Message	Notification	Calendar
Training Set	Shopping	5715	621	0
	Travel	2982	353	77
	Event	2609	467	217
	Service	2508	571	0
	Personal	3865	5309	235
Test Set	All five	2000	500	500

Table 2: Data Distribution Across All Domains

data. We then filtered the collected data to get $\sim 17k$ $(16,959)$ relevant messages, $\sim 7K$ (7321) relevant notifications and 720 calendar data and classified them into 5 domains (Shopping, Travel, Event, Service, Personal). The data distribution over these domains is shown in Table 2.

The collected data was then clustered based on similar templates for the ease of annotation. For e.g. the product delivery messages from Amazon follows a certain template with only change being in certain fields such as product name, delivery agent contact etc. We then chose an exemplar from each of these clusters and asked annotators to annotate each of the words in the text with the appropriate tag. We then curated a test set from $\sim 9k$ relevant data collected from different individuals. This was kept separate from the training data and was handpicked to include unique instances.

Table 1 in appendix displays one sample instance per domain and the identified relevant attributes while Table 2 covers the entity linker output for the previously chosen sample instances along with relevant fields like Sender and Date. The list of all relevant attributes was generated by a comprehensive analysis of collected data and usefulness of information contained in the data source. The distribution and relevant domains for each attribute is given in Table 3.

Attribute	Relevant Domains	Count
ID	Shopping, Travel, Event, Service	3652
Status	Shopping, Travel, Event, Service	8751
Vendor	Shopping, Travel, Event, Service	3961
Product	Shopping	2910
Start Date	Travel, Event	2749
End Date	Shopping, Travel, Event, Service	2164
Start Time	Travel, Event, Service	2913
End Time	Travel, Event, Service	1458
Travel Mode	Travel	1835
Travel Class	Travel	840
PNR	Travel	1553
Vehicle Number	Travel	1829
Source Location	Travel	2117
Destination	Travel, Event	2402
Event Name	Event	639
Event Type	Event	131
Service Type	Service	143
Amount	Shopping, Travel	1203
Relationship	Personal	41
Occasion	Personal	76

Table 3: Distribution of Attributes and Relevant Domains

5 Experiments and Results

5.1 Evaluation Metrics

We performed evaluation on the test set (described in Table 2). We used weighted F1 score, precision and recall to evaluate performance of our proposed pipeline. Since our system focuses on on-device extraction, latency and memory usage are also critical metrics. The total model size (including embedding size) is also reported for every model. The latency measurements were done by averaging over randomly picked 100 data points.

5.2 Domain Classifier

We compare 3 different models for domain classification and the model parameters are given in Table 4.

- **Hybrid Naïve Bayes Classifier:** This is the proposed domain classifier. We computed Tf-Idf of unigram, bigram, trigram and quadgram tokens and used these to compute class probabilities.
- **Deep Neural Network (DNN):** We generate token embeddings using a filtered version of Glove embeddings to reduce memory usage. The model accepts these token embeddings as input and generates a distribution over the predefined set of domains.
- **Convolutional Neural Network (CNN):** Like in the previous model, we generate token embeddings using filtered version of Glove embeddings. These token embeddings act as input for convolutional layers that extract feature maps. This is further passed into Max-pooling layer and then a final linear layer which gives output distribution.

From Table 5, we can see that all the models gave comparably high F1 scores on the test data with the Naïve Bayes classifier just edging the other two. Contrary to expectations, the deep learning based approaches do not outperform the simpler Naïve Bayes model. This is because the domains do not overlap and have very distinct samples and hence, do not need a complex model for accurate distinction. Since all computed latencies are very low, we give preference to memory usage during selection.

5.3 Attribute Extractor

We compare the performance of 3 deep learning models with different encoders followed by a CRF decoder.

- **Bi-LSTM + CRF:** This is the standard approach for sequence labelling. The Bi-LSTM encodes the input and the CRF acts as the decoder. This model uses only the word level features of the sentence as input.

DNN	Number of layers	2
	Number of Hidden Units	50, 20
	Dropout value	0.5
CNN	Number of 2D convolutional layers	2
	Number of filters	64, 32
	Kernel dimensions	5, 3
	Dropout value	0.5
Optimizer & learning rate		Adam, 0.001
Train - validation split		80 - 20

Table 4: Domain Classifier Model Parameters

Metric	Näive Bayes	CNN	DNN
F1 Score	**0.9596**	0.9551	0.9561
Precision	**0.9713**	0.9472	0.9487
Recall	0.9482	0.9632	**0.9637**
Model Size (KB)	2680	378	983
Embedding Size (KB)	NA	4636	4636
Total Memory (KB)	**2680**	5014	5619
Latency (ms)	91.7	**14.7**	19.8

Table 5: Domain Classifier Results

- **Dual Bi-LSTM + CRF:** This is the proposed model. It captures the character level information along with word level features.
- **Transformer + CRF:** We use a multi-headed transformer encoder followed by a CRF decoder.

The model details for each aforementioned approach are given in Table 6. We also experiment addition of the domain classifier output as input into the attribute extraction model. The domain is added as an extra input to the message/notification and then the combined input is fed into the embedding layer.

We see from Table 7 that our proposed Dual Bi-LSTM encoder just outperforms the standard Bi-LSTM. This verifies the ability of character embeddings to capture greater morphological diversity, which is especially visible in SMS and notifications. We also experiment with the transformer model for the attribute extraction task. Owing to the huge model size of pre-trained transformers like BERT, we limit ourselves to a custom two layer transformer model which is trained from scratch. The inferior performance of the transformer model compared to the Bi-LSTM model can be attributed to over-parameterization and lack of pre-training. We also observe that adding domain input slightly worsens the performance. This is because our attributes are structured such that similar attributes across different domains are considered as one. E.g. Delivery Date, Event Date and Service Date are all considered as End Date. Hence, adding the domain input adds to the complexity of input and we observe slight drop in performance.

We also run our models on the popular English NER dataset - CoNLL2003. It contains four different named entities: PERSON, LOCATION, ORGANIZATION, and MISC. The dataset consists of 14000 training samples, 3200 validation samples and 3500 test samples. The data is tokenized using the same preprocessing as mentioned in 3.1.

BiLSTM + CRF	Number of layers	2
	Number of Hidden Units	128
	Embedding Dimension	128
Dual Bi-LSTM + CRF	Number of layers	2
	Number of hidden units	128
	Word embedding dimension	128
	Character embedding dimension	64
	Number of char-LSTM hidden units	64
Transformer + CRF	Number of layers	2
	Positional encoding dimension	128
	Embedding dimension	128
	Number of attention heads	4
	Bidirectional	true

Table 6: Attribute Extractor Model Parameters

Metric	Dual Bi-LSTM + CRF	Bi-LSTM + CRF	Transformer + CRF
F1 Score	**0.9329**	0.9308	0.9037
F1 Score (with domain)	**0.9311**	0.9289	0.8976
Precision	**0.9392**	0.9333	0.9132
Recall	0.9281	**0.9307**	0.9078
Model Size (MB)	6.4	**5.4**	27
Latency (ms)	77	**50**	116

Table 7: Attribute Extractor Results on our collected dataset

The results of our models on CoNLL2003 test set are given in Table 8. We see that our proposed model achieves reasonably high F1 score (within $\sim$ 1% of current state of the art). A significant advantage of our approach is the simplicity of the model which allows it to be deployed on-device as well. Unlike our proposed model that involves training embeddings from scratch, all models that outperform our proposed model make use of powerful pre-trained or contextualized embeddings. Another interesting trend we observe is the similar pattern of performance of our three models across both datasets. This verifies that our proposed model performs the best for on-device entity extraction.

Models	F1 Score
LSTM-CRF (Lample et al., 2016)	90.94
Bi-LSTM-CNN-CRF (Ma and Hovy, 2016)	91.22
LM-LSTM-CRF (Liu et al., 2017)	91.25
Transformer-CRF (Ours)	91.75
Bi-LSTM-CRF + ELMO (Peters et al., 2018)	92.2
TENER (Yan et al., 2019)	92.63
BERT (Devlin et al., 2019)	92.8
Bi-LSTM-CRF (Ours)	92.85
Flair (Akbik et al., 2018)	93.1
Dual Bi-LSTM-CRF (Ours)	93.28
Cross Weigh + Pooled Flair (Wang et al., 2019)	93.43
Baevski et al. (Baevski et al., 2019)	93.5
LUKE (Yamada et al., 2020)	94.3

Table 8: Benchmark results on CONLL-2003 dataset

5.4 Entity Linker

For the entity linker we achieve an F1 score of 0.98. This module has a very high F1 score because most of the entity linking is done using string-matching and phonetic algorithms and regex pattern matching as compared to earlier modules, which involved machine learning/ deep learning models. The model size for this module is 284KB and latency is 40 ms;

5.5 Engine Pipeline

The complete engine pipeline is implemented as a service on device. The attribute extractor is implemented in PyTorch and the trained model is converted to android (v10) compatible version using the Pytorch JIT module. On-device inference is done using PyTorch android runtime. Similarly, Domain classifier is implemented in tensorflow and on-device inference is done using tensorflow-lite android library. Trained models were tested and deployed on Samsung Galaxy S10 and Note 10 devices.

The final end to end pipeline consists of 4 different modules, Classifier, Attribute Extractor, Entity Linker & Triplet Builder. The respective F1 scores achieved on the test set for the 4 modules are 95.96, 93.29, 98 & 92.4 respectively. The end to end accuracy of the overall system is 81.06 %. The outputs of each of these individual modules is included in the appendix. After porting to the device the model and app sizes were recorded to be 8.31 MB & 48.87 MB respectively, and engine's latency was measured to be ~200 ms.

6 Conclusion

In this paper, we have proposed a unique on-device pipeline to extract relevant information from unstructured sources such as messages, notification, calendar entries etc. By making on-device extraction very efficient, we eliminate the issue of user privacy while providing a platform for an enhanced personalised experience. Since the relevant domains are majorly non-overlapping, a Naïve Bayes classifier gives sufficiently good performance. For attribute extraction, we propose a dual word and character based Bi-LSTM + CRF model, which achieves best results on our self-curated test set as well as CONLL-2003 test set.

The feasibility of such a system was claimed through an on-device implementation using the proposed approach. The applications of such an on-device system can be envisioned across various personalization and recommendation services while maintaining user privacy.

7 Future Work

A possible extension of this work is to extend the English information extraction system to a multilingual one. This is an interesting area of exploration because each language has a different morphology, so it will be more challenging for a single model to capture multilingual features. Currently, our system is a pipeline consisting of several models, which can cause propagation of error. So, exploring the possibility of an end-to-end information extraction system is another direction in which we can expand our research.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Tiago A. Almeida, José María G. Hidalgo, and Akebo Yamakami. 2011. Contributions to the study of sms spam filtering: New collection and results. In *Proceedings of the 11th ACM Symposium on Document Engineering*, DocEng '11, page 259–262, New York, NY, USA. Association for Computing Machinery.

Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. 2019. Cloze-driven pretraining of self-attention networks. 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 5360–5369, Hong Kong, China. Association for Computational Linguistics.

Richard Cooper, Sajjad Ali, and Chenlan Bi. 2005. Extracting information from short messages. In *Natural Language Processing and Information Systems*, pages 388–391, Berlin, Heidelberg. Springer Berlin Heidelberg.

Gordon V. Cormack, José María Gómez Hidalgo, and Enrique Puertas Sánz. 2007. Spam filtering for short messages. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, page 313–320, New York, NY, USA. Association for Computing Machinery.

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.

Fatia Kusuma Dewi, Mgs. M. Rizqi Fadhlurrahman, Mohamad Dwiyan Rahmanianto, and Rahmad Mahendra. 2017. Multiclass sms message categorization: Beyond spam binary classification. In *2017 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pages 210–215.

Tobias Ek, Camilla Kirkegaard, Håkan Jonsson, and Pierre Nugues. 2011. Named entity recognition for short text messages. *Procedia - Social and Behavioral Sciences*, 27:178–187. Computational Linguistics and Related Fields.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Huixing Jiang, Xiaojie Wang, and Jilei Tian. 2010. Second-order hmm for event extraction from short message. In *Proceedings of the Natural Language Processing and Information Systems, and 15th International Conference on Applications of Natural Language to Information Systems*, NLDB'10, page 149–156, Berlin, Heidelberg. Springer-Verlag.

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, page 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016.

Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Yuanchun Li, Ziyue Yang, Yao Guo, Xiangqun Chen, Yuvraj Agarwal, and Jason I. Hong. 2018. Automated extraction of personal knowledge from smartphone push notifications. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 733–742.

Liyuan Liu, Jingbo Shang, Frank F. Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2017. Empower sequence labeling with task-aware neural language model. *CoRR*, abs/1709.04109.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Matthew E. 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.

Joseph Polifroni, Imre Kiss, and Mark Adler. 2010. Bootstrapping named entity extraction for the creation of mobile services. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

S. Vatsal, N. Purre, S. Moharana, G. Ramena, and D. Mohanty. 2020. On-device information extraction from sms using hybrid hierarchical classification. In *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*, pages 178–181, Los Alamitos, CA, USA. IEEE Computer Society.

Zihan Wang, Jingbo Shang, Liyuan Liu, Lihao Lu, Jiacheng Liu, and Jiawei Han. 2019. CrossWeigh: Training named entity tagger from imperfect annotations. 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 5154–5163, Hong Kong, China. Association for Computational Linguistics.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, Online. Association for Computational Linguistics.

Hang Yan, Bocao Deng, Xiaonan Li, and Xipeng Qiu. 2019. Tener: Adapting transformer encoder for named entity recognition.

Weakly Supervised Extraction of Tasks from Text

Sachin Pawar, Girish K. Palshikar, Anindita Sinha Banerjee
TCS Research, Pune, India.
{sachin7.p,gk.palshikar,anindita.sinha2}@tcs.com

Abstract

In this paper, we propose a novel problem of automatic extraction of tasks from text. A task is a well-defined knowledge-based volitional action. We describe various characteristics of tasks as well as compare and contrast them with events. We propose two techniques for task extraction – i) using linguistic patterns and ii) using a BERT-based weakly supervised neural model. We evaluate our techniques with other competent baselines on 4 datasets from different domains. Overall, the BERT-based weakly supervised neural model generalizes better across multiple domains as compared to the purely linguistic patterns based approach.

1 Introduction

We define a *task* as a well-defined knowledge-based action with a specific goal and which is carried out within a small time period, often by a single person, a group of persons, a device, or a system. The tasks usually demand some skill and expertise by the human actor(s) performing the task and they are carried out volitionally by the actor(s).

The problem of *Task extraction* is to automatically identify mentions of such tasks in text. Syntactically, a task can be mentioned as a verb phrase (e.g., `implemented a model for weather prediction`) or as a noun phrase (`model implementation for weather prediction`) in a sentence. Table 1 shows various examples of tasks observed across multiple domains (also see Table 6 for a comprehensive list of tasks). The extent of a task mention should be such that the complete meaning expressed by the task should be captured. For example, from the sentence `The researcher implemented a model for weather prediction.`, it is expected to identify the entire phrase `implemented a model for weather prediction` as a task, even though the shorter phrase `implemented a model` is a valid task mention but does not capture the entire meaning.

Event extraction (Xiang and Wang, 2019) is a popular task in NLP literature. An *event* is generally defined as a specific occurrence of something happening in a certain time and place which involves one or more participants and can often be described as a change of state. Although events are similar to tasks in some aspects, there are certain crucial distinctions (described in detail in Section 2) and hence it is important to define and address task extraction as a separate problem.

There are several interesting analyses that can be carried out over the extracted tasks from large corpora. Similar tasks can be mentioned in different ways and it is important to cluster the tasks together which have similar *meanings*. By definition, a task needs certain expertise to be carried out and it would be an interesting problem to determine difficulty level for each task. Usually, each task is carried out by an actor and this actor often plays a certain generic *role* such as engineer, banker, farmer etc. Also, most tasks need certain skills to be carried out such as various technical or domain concepts, programming languages, certain tools or technologies, etc. Such co-occurrence or interdependence between tasks and roles as well as between tasks and various skills, can be studied.

Task extraction has several useful real-life applications. For example, tasks extracted from *resumes* capture the fine-grained experience of the candidate and would be quite useful for automatically shortlisting candidates for a certain job requirement. Another interesting application of the extracted tasks and their corresponding roles is to automatically augment common sense knowledge. For example, ConceptNet (Speer et al., 2017) contains knowledge of the form ⟨ `Engineer`; *is capable of*; building a bridge⟩ or ⟨ `Policeman`; *is capable of*; arresting criminals⟩. But the number of such triplets are lim-

Dataset	Type	Examples of Tasks
Resumes	Verbal	We compared the techniques with which **[low power cascade amplifiers can be developed].**
	Nominal	Verified the credibility of GA algorithm in **[designing of orthogonal waveform for MIMO radar].**
TechCrunch	Verbal	But Kaliszan and Robertson realized that commercial security was so backward that just **[implementing the established principles of machine vision]** and the cloud could create a huge company.
	Nominal	Remote work is what led to **[the development of GitLab's publicly viewable handbook].**
Patents	Verbal	Particularly, **[a new e-mail message is created for each software package to be deployed] .**
	Nominal	The computer implemented method includes performing, using a processor, **[a static timing analysis of the integrated circuit].**
Reuters News	Verbal	At noon the bank had **[estimated the shortfall at 500 mln stg].**
	Nominal	Stock market analysts said today's generally weak stock market plus unwinding of positions after **[heavy buying of BAT shares in the run-up to the results]** caused the fall in the share price.

Table 1: Examples of Tasks mentioned in various datasets. The task phrases are enclosed in square brackets and are highlighted in bold within sentences. "Type" indicates the syntactic type of the task phrase which depends on the POS tag of the head word of the phrase – verbal (e.g., `developed`) or nominal (e.g., `development`).

ited to a very few tasks. Using tasks extracted from a large corpus, we can automatically augment such common sense knowledge.

In this paper, we focus on the problem of automatic extraction of tasks from text and propose two techniques for that. The first technique makes use of linguistic patterns and resources such as WordNet (Miller, 1995). The second technique uses a weakly supervised BERT-based neural model and employs the Snorkel framework (Ratner et al., 2017) for automatically creating a labeled training dataset. The rest of the paper is organized as follows – Section 2 discusses relevant past work. Section 3 describes the two proposed techniques for the automatic extraction of tasks from text. Section 4 describes detailed experimental analysis including dataset descriptions, baselines, and evaluation. Finally, We conclude in Section 5 along with some potential future work.

2 Related Work

To the best of our knowledge, ours is the first attempt to introduce the problem of task extraction and propose extraction techniques for it. Event extraction (Xiang and Wang, 2019) is a related but different problem as compared to task extraction. Definition of an *event* varies depending on its application and context. Sims et al. (Sims et al., 2019) have defined an *event* to be what is depicted as actually occurring in text (also referred as *realis events*). Such events are expected to have four important aspects (non-negation, tense, genericity, and modality) which we compare and contrast with tasks in the top four rows of Table 2. We also describe two more important aspects of tasks – *volitionality* and *need for expertise*. Overall, although there is overlap between tasks and events, neither is an subset of the other.

3 Task Extraction

In this section, we discuss the two techniques for extraction of tasks from text.

3.1 Using Linguistic Patterns

Linguistic patterns for task extraction do not need any training data. We define an *action noun* as a noun that indicates an action, activity etc.; e.g., `improvement`, `design`, `review`, `selection`, `administration`. A noun is accepted as an action noun if (i) its hypernym tree (for any of its top $k_0 = 2$ senses) includes *action indicator words* like `work`, `activity` , `human_action`, `group_action` etc. (e.g., hypernym tree for sense 2 of `improvement` is: *change of state* $\rightarrow$ *change* $\rightarrow$ *action* $\rightarrow$ ***human_action***); or (ii) Alternatively, the noun's category in WordNet should be `noun.act`. Generally, we do not consider abstract nouns (e.g., `idea`) as action nouns; and (iii) the noun is not present in a domain-specific negative list; for IT domain, some examples of negative action nouns are: `project`, `technology`, `job`, `approach`, `practice`, `procedure` etc.

Similarly, we accept a verb V to be an action verb[1] if: (i) one of the derivationally re-

[1]The list of action nouns and verbs which is created using WordNet, will be made available upon request.

Aspect	Description
Non-negation	Events should be explicitly mentioned as having occurred; typically there is no direct negation used to describe an event. Tasks are also similar to events in this aspect. In the sentence `He did not implement the solution`, neither an event nor a task is mentioned.
Tense	Events must be in the past or present tense. Tasks need not only be in past or present tense, they can be mentioned in future tense as well. In the sentence `He will implement the solution`, a task `implement the solution` is mentioned but it is not an event.
Genericity	Generic events (`Engineers build bridges`) describe a general category as against specific events which describe a specific occurrence (`L&T engineers built this bridge`). In event extraction, only specific events are considered whereas tasks can be generic events.
Modality	Only *realis* events which have *actually occurred* are considered as events. All other modalities such as *belief*, *hypothesis*, *desire* are not considered events but these can be considered as tasks. In the sentence `Engineers are supposed to build bridges which last for years`, a task `build bridges` is mentioned but it is not an event.
Volitionality	Tasks are those actions which are carried out by the actor *volitionally*. For example, `The bridge collapsed` is an event but it is not a task because the action of `collapsing` is not carried out volitionally by any actor.
Expertise	Unlike events, we define tasks as those actions which need some domain expertise or knowledge by the actor for execution. For example, `John entered the restaurant` is an event but not a task.

Table 2: Comparing *Events* and *Tasks* based on various aspects

lated nominal forms of V is an action noun; or (ii) the verb category in WordNet is any of: `verb.change`, `verb.motion`, `verb.creation`, `verb.social`, `verb.communication` etc. For example, there are 2 nouns related to sense 1 of the verb `provide`, namely, `provision` and `provider`. The second sense of `provision` includes `activity`, making it an action noun and hence `provide` is an action verb. Other examples: `improve`, `stabilize`, `plan`, `control`, etc.

Examples of linguistic rules to identify tasks having different syntactic structures are as follows:
• A noun compound (i.e., a sequence of nouns) is a task if the last noun is an action noun. Examples: `process improvement`, `user interface design`, `version control`, `asset management`
• An action verb in simple present/past tense or in gerund form and its direct object noun phrase (NP) form a task. Examples: `manage data center`, `implement quality processes`, `managing attrition`, `maintaining financial discipline`, `resolved customer complaints`
• An action verb V in gerund form connected to a preposition p using dependency relation (DR) *prep* followed by an NP connected to p using DR *pobj* is a task. Examples: `managing of large teams for customer support`, `coordinating with various vendors`
• Same as above, except instead of V a noun compound headed by an action noun is required. Examples: `analysis of existing bugs`, `Eigenvalue computation by application of numerical computation techniques`

3.2 Weakly Supervised BERT-based Task Extraction

The linguistic patterns based approach has certain limitations which need to be addressed to further improve the task extraction accuracy. The patterns check for the presence of action verbs and nouns and then extract their entire verb or noun phrases as tasks. However, the presence of action verbs or nouns is just a *necessary* condition and not a *sufficient* condition for being tasks. Two important aspects of tasks *volitionality* and *need for expertise* are not checked explicitly. Moreover, there is a challenge of polysemy which is not handled explicitly. A verb (or noun) may be an action verb (or an action noun) in one particular sense but may not be an action verb (or action noun) in another sense. For example, `synthetic data generation` is a valid task but `next generation of chip technology` is not a valid task because of different senses of the noun `generation`.

To overcome the above-mentioned limitations of our linguistic patterns based approach, we propose to learn a classification model which predicts whether any noun or verb in a sentence represents a *head word* of a valid task phrase. Linguistically, the head word of a phrase is the word which determines the syntactic category (e.g., noun phrase, verb phrase) of the phrase. We use the following definition of a head word considering the dependency parse tree – the head word of a phrase is syntactically the most important word in the phrase which connects it to the rest of the sentence, all other words in the phrase are directly or indirectly dependent on the head word. Once we identify head words of task phrases, we use the dependency tree structure of the sentence to get the corresponding complete task phrase. The rules for phrase expansion are described later in detail.

3.2.1 Classification Problem

Input: A word w in sentence S
Output: Predict one of the two class labels - TASK and NOT-TASK indicating whether or not the word w is a head word of a valid task phrase

3.2.2 Training Data

As there is no prior work for addressing this problem of extracting tasks, there are no readily available annotated datasets which we can use for training the above-mentioned classification model. Hence, we use the Snorkel framework (Ratner et al., 2017) for rapidly and automatically creating labeled training data. Snorkel enables writing multiple *labeling functions* (LFs) where each LF expresses an arbitrary heuristic for class label predictions. These LFs can have unknown accuracies and correlations but Snorkel denoises their outputs and combines their predictions to arrive at a final probability distribution over labels for each instance. A large training set can then be constructed rapidly using these automatically assigned *soft* labels (because of a probability distribution over labels and not a single hard label for each instance) and this training data can be used to train a machine learning model. We designed several LFs that capture various linguistic characteristics which we expect to be present in tasks.

3.2.3 Labeling Functions

We designed the following LFs where each LF assigns TASK or NOT-TASK for a classification instance. An LF need not assign labels for all instances, it may ABSTAIN for certain instances where it is unsure. A classification instance is a combination of a word w, the corresponding sentence S, and the dependency tree DT of S.

Action verbs or nouns: If the word w is not an action verb or noun (as per the list of action verbs and nouns prepared using WordNet in Section 3.1) then it is NOT-TASK. Here, the sentence context is not used and the decision is only based on the word w. E.g., nouns such as `book`, `culture` and verbs such as `situate`, `lack` are not tasks. All the subsequent LFs also predict NOT-TASK for non-action verbs and nouns but they also predict TASK or NOT-TASK for action nouns and verbs provided certain other conditions are satisfied.

Negation modifier: If the word w is modified by any negation indicating word (e.g., `not`, `never`) through dependency relation neg in DT then it is NOT-TASK. E.g., `They did not develop any weather prediction model`[2]. We also consider other ways of expressing negation such as `failed to develop` or `absence of development`.

Animate or organization agent: If the agent (dependency child with relation $nsubj$ or $agent$) of the verb w is *animate* or corresponds to some organization, then it is TASK. This LF captures volitionality in an implicit way as the animate agents (or organizations) indicate that the action corresponding to verb w is likely to be carried out volitionally. Here, animate/organization agents are those words which are – i) personal pronouns like `he`, `she`, `we`, ii) named entities of type PERSON or ORGANIZATION, or iii) person or organization indicating common nouns like `engineer`, `farmer`, `department` (these words have `person` or `organization` as their ancestors in WordNet hypernym tree). E.g., `Any overseas data demands are screened by the` **`department`**`.`

Inanimate agent: If the agent of the verb w is *inanimate*, then it is NOT-TASK. This LF captures the opposite characteristics as compared to the previous LF. The heuristic is that if any action is carried by an inanimate agent then it is unlikely to be a task. Here, inanimate agents are those words which are – i) event indicating nouns (e.g., `storm`, `pandemic`) or ii) natural objects or substances (e.g., `stone`, `water`). Again, lists of such words are created using WordNet hypernym structure. E.g., `The coronavirus` **`pandemic`** `accelerated the shift to e-commerce.`

Volition marker: If the verb w is modified by an adverb (dependency child with relation $advmod$) explicitly indicating volition, then it is TASK. Examples of volition indicating adverbs are `deliberately`, `voluntarily`, `intentionally`. E.g., `He` **`voluntarily`** `engages in self-developmental activities.`

Non-volition marker: If the verb w is modified by an adverb explicitly indicating non-volition, then it is NOT-TASK. Examples of non-volition indicating adverbs are `accidentally`, `unintentionally`. E.g., `He` **`accidentally`** `pressed the send button.`

[2]The word w is underlined and the same convention is followed for subsequent examples

Explicit expertise marker: If the word w occurs in explicit *expertise indicating context* in the dependency tree DT, then it is TASK. One of the key aspect of a task is that it needs certain domain expertise to be executed. Expertise indicating context can be – i) w being modified by an adjectival clause headed by `using` or `leveraging`, or ii) w being preposition phrase modifying nouns/verbs such as `knowledge of` or `expertise in`. E.g., `You can dynamically deliver language files to your mobile apps` **`using`** `SDKs.`

Expertise score using corpus statistics: If the word w has high *expertise score* based on corpus statistics then it is TASK. This LF does not use the sentence context for the word w. Here, we compute expertise score for each action noun and verb using statistics from a large corpus; and then choose top 100 action verbs and nouns using this expertise score. We used 3.6 million sentences from ukWaC, a very large web-derived corpus of English (Ferraresi et al., 2008). For each action verb and noun, the expertise score is computed as follows:

$$ExpScore(w) = log(N^e_w + 1) \times \frac{N^e_w}{N_w} \qquad (1)$$

Where, N_w: No. of times the word w appears in the corpus and N^e_w: No. of times the word w appears in the explicit *expertise indicating context* (as described in the previous LF). The score for a word will be high if both of the following conditions are true – i) the conditional probability of observing it in expertise indicating context is high, and ii) the absolute frequency with which it appears in expertise indicating context is high. This is motivated by the patterns scoring formula used by Thelen and Riloff (2002). E.g., `The Fosters will develop media like podcasts and videos.` and `The work involves experimental investigation of droplet impingement over a heated surface.`

Presence of direct object: If the word w has a direct object (*dobj*) or a passive subject (*nsubjpass*) then it is TASK. For actions expressed using nouns, prepositional phrase headed by `of` is considered similar to a direct object (`implemented the solution` $\Rightarrow$ `implementation of the solution`). Here, the heuristic is that the presence of direct object (or passive subject for passive voice verbs) for an action verb increases likelihood of it being a more meaningful task. E.g., `It recently published a` **`handbook`**`.`; `The` **`post`** `was restricted on social media.`; and `It asks user for selection of a particular` **`webpage`**`.`

Absence of direct object or prepositional modifier: If the verb w does not have any direct object, passive subject, or any prepositional modifier, then it is NOT-TASK. This LF captures the opposite characteristic as compared to the previous LF, with the heuristic that such verbs are unlikely to constitute meaningful tasks. E.g., `It allows the VM to begin operating quickly.`

Adjectival clause modifier: If the verb w is a head word of an adjectival clause modifying some noun, then it is NOT-TASK. Here, the heuristic is that such verbs simply provide extra information about a noun and are unlikely to be tasks. E.g., `It left thousands of sensitive health records exposed to the internet.`

Compound noun modifier: If the noun w modifies another noun as a compound modifier, then it is NOT-TASK. E.g., `Rivian sets a delivery date.`

Number-like modifier: If the noun w is modified by a number, an ordinal, a cardinal, or a number-like modifier like `next`, then it is NOT-TASK. E.g., `The solutions that the first generation of clean tech investors backed were economically unfeasible.`

3.2.4 BERT-based classification model

We propose a BERT-based (Devlin et al., 2018) classification model which predicts an appropriate class label (TASK vs NOT-TASK) for each word in a sentence. The annotated data needed for training this model is created automatically using the Snorkel framework with the labeling functions described above. Each instance is annotated with soft labels, i.e., a probability distribution $y_{gold} \in \mathbb{R}^2$ over TASK and NOT-TASK. Each instance is a combination of a word w, its POS tag p, and the complete sentence S.

We now describe the classification model in detail. Figure 1 depicts the model architecture. First, embedded representation $x_w \in \mathbb{R}^{768}$ is obtained for the word w using a pre-trained BERT transformer model.

$$x_w = BERT(S, w) \qquad (2)$$

We then use a linear feed-forward layer to get a more compressed representation $x'_w \in \mathbb{R}^{10}$ of the word.

$$x'_w = ReLU(Wx + b) \qquad (3)$$

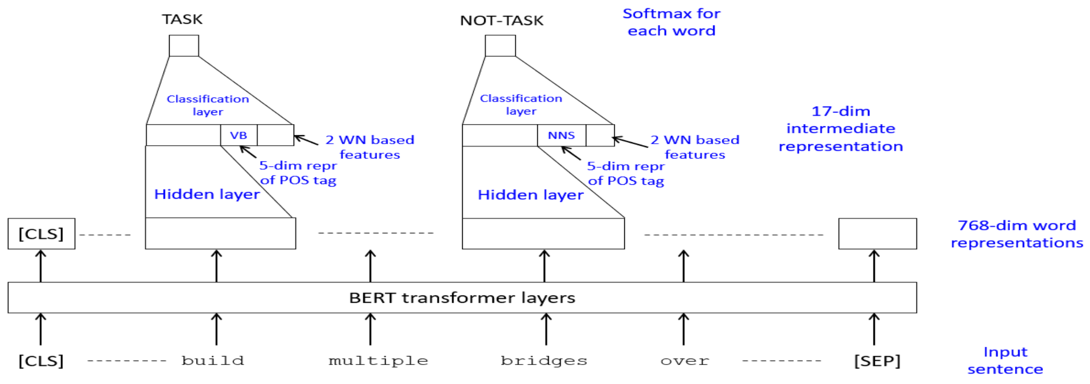


Figure 1: Architecture of BERT-based Task Extraction model

Here, $W \in \mathbb{R}^{10\times768}$ and $b \in \mathbb{R}^{10}$ are learnable weights. We compress the word representation to lower dimensions because we use 3 additional features described below and we do not want the BERT-based features to overwhelm these additional features.

POS tag: Part of speech tag of the word w is a key feature because the head word of a task can only be a noun or verb. We use embedded representation $x_p \in \mathbb{R}^5$ of the POS tag of w using an Embedding layer with learnable weights ($\mathbb{R}^{N_p\times5}$ where N_p is the number of distinct POS tags).

WordNet-based features: We hypothesize that more *specific* words are more likely to be task head words than *generic* words. Hence, we use two WordNet-based features to implicitly estimate *specificity* of the word w – i) Hypernym depth and ii) Corpus frequency. Hypernym depth of a word is the number of levels in the hypernym tree between that word and the root (`entity` in case of all nouns). The higher the hypernym depth, the higher is its specificity. E.g., hypernym depth of `diagnosis` is 8 which is higher as compared to a more generic word `action` with hypernym depth of 5. Similarly, lower the corpus frequency of a word, higher is its specificity. We use the corpus frequencies provided for each lemma in each synset in WordNet (Jurafsky and Martin, 2021). E.g., corpus frequency of `see` is 613 but for `analyze` it is only 21 which is more specific word. For each word, we use WordNet-based features vector $x_{wn} \in \mathbb{R}^2$

Overall representation of the word w, $h_w \in \mathbb{R}^{17}$ is now concatenation of x'_w, x_p, and x_{wn}. This is then passed through the final classification layer to get the predicted label distribution $y_{pred} \in \mathbb{R}^2$.

$$h_w = Concatenate([x'_w; x_p; x_{wn}]) \quad (4)$$

$$y_{pred} = Softmax(W'h_w + b') \quad (5)$$

$$loss = KLDLoss(y_{gold}, y_{pred}) \quad (6)$$

Here, $W' \in \mathbb{R}^{2\times17}$ and $b' \in \mathbb{R}^2$ are learnable weights. The predicted label distribution is then compared with the gold standard or expected distribution y_{gold} to compute KL divergence loss which is back-propagated during the training process. We also fine-tune the final encoder layer of BERT.

3.2.5 Phrase Expansion

The classification model identifies task head words which need to be expanded to get complete task phrases. We use a few simple rules to expand head words to phrases using the dependency tree of the sentence. Basically, we need to get the phrase corresponding to the dependency subtree rooted at the head word but we need to discard certain dependencies. We recursively collect a set of dependency children starting from the head word and construct the phrase from the leftmost child to the rightmost child. However, we do not consider dependency child connected to its parent with certain dependency relations and hence do not recurse on such children further. Dependency relations which we discard are – i) $nsubj$, $agent$ (because task phrase does not contain the agent who executed the task); ii) $relcl$, $advcl$, $ccomp$, $appos$ (to avoid getting complete dependent clauses or appositives describing extra information inside a task phrase); iii) aux, $auxpass$ (for not including auxiliary verbs in task phrases). E.g., consider the task head word `analyzed` from the sentence in Figure 2. Here, the expanded task phrase is `analyzed traffic at internet exchanges`. Here, `firm` is excluded because its dependency relation $nsubj$ is discarded, but other dependency children of `analyzed` are included recursively.

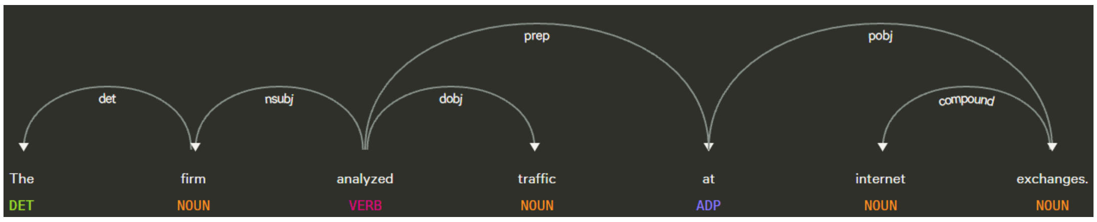


Figure 2: Example of a dependency tree and phrase expansion for the task head word `analyzed`. (Courtesy: spaCy dependency visualizer at `https://explosion.ai/demos/displacy`)

4 Experimental Analysis

In this section, we describe our experiments in detail including datasets, baselines, evaluation metrics and results.

4.1 Datasets

In order to evaluate the task extraction performance, we chose 4 datasets from different domains.
• **Resumes**[3]: Set of resumes of candidates shared with our organization
• **TechCrunch**[4]: Articles from technical domain published on TechCrunch in 2020
• **Reuters**: A collection of 10,788 documents from the Reuters financial newswire service from the well-known Reuters-21578 corpus (Lewis, 1997).
• **Patents**[5]: Abstracts of patents assigned to IBM in the years 2000-2019 which are scraped from Google Patents

4.1.1 Training Data

The training data for our BERT-based task extraction model is generated automatically using the Snorkel framework based on several labeling functions described in Section 3.2.3. We randomly chose 1000 documents from the 4 datasets (250 documents from each) – Resumes, TechCrunch, Reuters, and Patents. The dataset consists of 19268 sentences where 98044 words (verbs and nouns) are assigned soft labels using Snorkel's labeling model (Ratner et al., 2017) which combines predictions of our labeling functions. Out of 98044 words, 21892 words were labeled as likely TASK head words, i.e., they were assigned TASK probability greater than 0.5. For all the remaining words in these sentences, NOT-TASK is considered as a hard label. Table 3 shows various statistics of these labeling functions on this training data. Detailed hyper-parameter details used for training this model are inlcuded in the Appendix.

[3]This is an internal dataset and can not be made public due to privacy reasons.

[4]`https://www.kaggle.com/sumantindurkhya/techarticles2020`

[5]`https://www.kaggle.com/federicolusiani/ibm-patents`

Labeling function	Cov	Overlap	Conflict
non action nouns/verbs	0.625	0.625	0.000
negation modifier	0.630	0.629	0.006
animate/org agent	0.680	0.667	0.021
non-animate agent	0.638	0.634	0.010
volition marker	0.625	0.625	0.000
non-volition marker	0.625	0.625	0.000
explicit expertise marker	0.629	0.629	0.001
corpus expertise score	0.673	0.660	0.013
direct object	0.799	0.687	0.013
adjectival clause	0.641	0.630	0.002
no object or pp	0.708	0.651	0.029
compound modifier	0.641	0.625	0.000
number-like modifier	0.625	0.625	0.000

Table 3: Analysis of labeling functions over our training dataset (Cov: Fraction of training instances labeled (not abstained) by the LF, Overlap: Fraction of training instances where the LF has predicted along with at least one other LF, Conflict: Percentage of overlapping training instances where there is mismatch of predicted label with at least one other LF)

Dataset	#Sentences	#Tasks
Resumes	1297	167
TechCrunch	292	251
Reuters	178	89
Patents	102	100
Total	1869	607

Table 4: Details of the evaluation dataset

4.1.2 Ground Truth

In order to create ground truth for evaluating various task extraction techniques, we manually annotated 20 documents from each of the 4 datasets with gold-standard task head words and complete task phrases. This dataset[6] consists of 1869 sentences where 607 tasks are annotated (see Table 4).

[6]Evaluation dataset as well as automatically labeled training dataset (excluding Resumes) will be made available upon request.

Dataset	Technique	Only Task head word			Lenient evaluation			Strict evaluation		
		P	R	F1	P	R	F1	P	R	F1
Resumes	EvExtB1	0.553	0.166	0.255	0.380	0.103	0.162	0.314	0.086	0.136
	EvExtB2	0.335	0.669	0.447	0.373	0.714	0.49	0.232	0.454	0.307
	Linguistic Patterns	0.582	0.771	**0.663**	0.551	0.730	**0.628**	0.429	0.568	**0.488**
	BERT Extractor	0.552	0.675	0.607	0.505	0.589	0.544	0.311	0.368	0.337
TechCrunch	EvExtB1	0.354	0.222	0.273	0.343	0.229	0.274	0.217	0.144	0.173
	EvExtB2	0.312	0.763	0.442	0.294	0.734	0.419	0.187	0.476	0.268
	Linguistic Patterns	0.404	0.510	0.451	0.420	0.542	0.473	0.239	0.310	0.270
	BERT Extractor	0.449	0.732	**0.556**	0.422	0.694	**0.524**	0.262	0.439	**0.328**
Reuters	EvExtB1	0.323	0.370	0.345	0.294	0.364	0.325	0.139	0.170	0.153
	EvExtB2	0.188	0.716	0.297	0.188	0.761	0.302	0.095	0.386	0.152
	Linguistic Patterns	0.210	0.358	0.265	0.218	0.364	0.272	0.122	0.205	0.153
	BERT Extractor	0.314	0.716	**0.436**	0.296	0.682	**0.412**	0.161	0.375	**0.225**
Patents	EvExtB1	0.533	0.075	0.132	0.556	0.085	0.148	0.267	0.034	0.061
	EvExtB2	0.371	0.774	0.502	0.370	0.752	0.496	0.179	0.385	0.244
	Linguistic Patterns	0.420	0.472	0.444	0.515	0.590	0.550	0.220	0.248	0.233
	BERT Extractor	0.524	0.830	**0.642**	0.522	0.803	**0.633**	0.268	0.419	**0.327**
Average	EvExtB1	0.441	0.208	0.251	0.393	0.195	0.227	0.234	0.109	0.131
	EvExtB2	0.302	0.731	0.422	0.306	0.740	0.427	0.173	0.425	0.243
	Linguistic Patterns	0.404	0.528	0.456	0.426	0.557	0.481	0.253	0.333	0.286
	BERT Extractor	0.460	0.738	**0.560**	0.436	0.692	**0.528**	0.251	0.400	**0.304**

Table 5: Comparative task extraction performance of our proposed techniques Linguistic Patterns and Weakly supervised BERT-based Task Extractor

4.2 Baselines

We consider two recent event extraction techniques as baselines for comparing the performance of our task extraction techniques.

EvExtB1: The first baseline is literary event extraction technique proposed by Sim et al. (2019). It is trained on literature dataset using a BiLSTM based model which used BERT token representations.

EvExtB2: The second baseline is an Open Domain Event Extraction technique proposed by Araki and Mitamura (2018). This is a more competent baseline because the events are not restricted to a domain or a syntactic type. It uses a BiLSTM based supervised event detection model which is trained on distantly generated training data.

For both the baselines, we use pre-trained models provided by the authors. Both the baselines identify event triggers that are considered as task head words and complete task phrases are identified using the phrase expansion rules described in Section 3.2.5.

4.3 Evaluation Metrics

Any gold-standard task phrase is counted as a true positive (TP) if there is a "matching" predicted task, otherwise it is counted as a false negative (FN). Here, two task phrases are considered to be "matching" if there is at least 80% string similarity between them for *strict evaluation* and 50% for *lenient evaluation*. All the remaining predicted tasks which are not TPs, are counted as false positives (FP). In addition, similar to event triggers, we also compute TPs, FPs and FNs considering only task head words. Precision, recall and F1-score are then computed for each of these three evaluation strategies – strict evaluation, lenient evaluation and considering only task head words.

$$P = \frac{TP}{TP + FP}; R = \frac{TP}{TP + FN}; F1 = \frac{2 \cdot P \cdot R}{P + R} \quad (7)$$

4.4 Results

We evaluate our proposed techniques – linguistic patterns and weakly supervised BERT-based task extractor, on 4 different datasets using 3 evaluation strategies. We compare the performance of the proposed techniques with two event extraction baselines. Table 5 shows the detailed results. Except the Resumes dataset, the BERT-based task extractor outperforms all other techniques on all the datasets. Considering the macro-average across datasets, the BERT-based task extractor turns out to be the best overall technique which also performs consistently across datasets. We also carried out ablation analysis to evaluate contribution of POS tag and WordNet-based features and observed that these features have minor positive contribution. Table 6 shows a comprehensive list of tasks extracted by the BERT-based task extractor from the 4 datasets.

Resumes:
-weld the P91 steel plate
-reduce the surface area of radiators
-develop a FEM code of multidimensional small deformation plasticity
-Cold flow analysis of double ramp flame holder
-Designed and analyzed a two stage worm gear box
-facilitate cab services
-Implemented a dynamic memory allocator for C
-Comparison of Different Clustering Techniques
-the generation of layered NAND gate
-Development of desktop application using deep learning
TechCrunch:
-enhance and improve antitrust regulations on the platform economy
-background data - mining of internet users
-tackling abusive behavior
-ensuring fairness in digital marketplaces
-verifying any system weaknesses and functioning of devices
-load up prior versions of iOS
-fire up a simulated iPhone and hunt for potential bugs
-discover potential security bugs
-spin up a virtualized ARM device (including iOS devices) in a browser
-filed a lawsuit against the virtualization software company
Reuters:
-facilitate a transaction
-fixed the value of the new Cruzado currency
-studying financially superior alternatives
-A bill also was introduced
-monitor coffee imports
-control the spread of the disease
-the susceptible clones would be replaced
-the disease was detected in nurseries
-restore normal moisture to the cane
-made appropriate declarations at customs points
Patents:
-a log transfer to the standby machine is performed
-the executing program is blocked
-the spatial index may be partitioned
-partition a spatial index into a plurality of portions
-A set of congestion cost corresponding to the set of pattern routes is computed
-data processing
-verification of a digital cir cuit design
-disseminate the write request
-performing multimodal analysis on the multimedia stream
-maintain the PIN diode bias as high as possible

Table 6: Examples of Tasks extracted from various datasets

4.5 Implementation Details

In this section, we describe the hyper-parameters used for training our BERT-based Tasks Extraction model. We have not carried out extensive hyper-parameter tuning but we set aside a small subset of training set as validation set, tried a few settings and chose the best one. We then re-trained our model using the entire training set with this set of hyper-parameters: batch size = 16 sentences, maximum sentence length = 128 tokens, number of epochs = 2, Adam optimizer with learning rate = 0.001. For avoiding overfitting, we used a dropout of 0.4 probability over the 768 dimensional token representation output by BERT. We also used gradient clipping to keep maximum norm of the gradient vector below 5. We used pre-trained base model of BERT from HuggingFace: `bert-base-uncased`[7]. While training our model, we also fine-tuned the last encoder layer of BERT (`encoder.layer.11`) and kept other layers' parameters frozen.

5 Conclusion and Future Work

In this paper, we have introduced a new NLP task of automatic extraction of *tasks* from text, highlighted the motivation behind it, and its potential applications. We described various aspects of tasks and highlighted how they compare with another popular NLP task of event extraction. We proposed two techniques for task extraction – i) linguistic patterns and ii) BERT-based weakly supervised neural model. We demonstrated effectiveness of our techniques on 4 datasets from different domains and compared them with other competent baselines. Given that ours is the first attempt for extracting tasks from text and the approach is only weakly supervised and does not demand any heavy manual annotation efforts, the overall performance is encouraging. However, there is still scope for the improvement which we plan to pursue as a future work. Also, we wish to refine the labeling functions to better capture linguistic characteristics of tasks such as volitionality, need for expertise, and execution within a small time period.

Acknowledgments

We thank the anonymous reviewers for their comments. We also thank our colleagues Nitin Ramrakhiyani and Dr. Swapnil Hingmire for helping to set up the event extraction baselines.

References

Jun Araki and Teruko Mitamura. 2018. Open-Domain Event Detection using Distant Supervision. In *Proceedings of the 27th International Conference on*

[7]https://huggingface.co/bert-base-uncased

Computational Linguistics (COLING), pages 878–891, Santa Fe, NM, USA.

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*.

Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and evaluating ukwac, a very large web-derived corpus of english. In *Proceedings of the 4th Web as Corpus Workshop (WAC-4) Can we beat Google*, pages 47–54.

Dan Jurafsky and James Martin. 2021. *Speech & language processing*. https://web.stanford.edu/ jurafsky/slp3/.

David Lewis. 1997. Reuters-21578 text categorization test collection, distribution 1.0.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

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.

Matthew Sims, Jong Ho Park, and David Bamman. 2019. Literary event detection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3623–3634.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence*.

Michael Thelen and Ellen Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the 2002 conference on empirical methods in natural language processing (EMNLP 2002)*, pages 214–221.

Wei Xiang and Bang Wang. 2019. A survey of event extraction from text. *IEEE Access*, 7:173111–173137.

A Resources created using WordNet and ukWaC corpus

As described in Sections 3.1 and 3.2.3, we have created several key resources using WordNet hypernym structure which includes lists of – action verbs, action nouns, person indicating nouns, organization indicating nouns, natural objects, and substances. We also computed expertise scores for various action nouns and verbs using a subset of ukWaC corpus. In our labeling function, we use top 100 action nouns and verbs as per this score. These lists which are created using WordNet and ukWaC corpus, will be made available upon request.

A German Corpus of Reflective Sentences

Veronika Solopova[*], **Oana-Iuliana Popescu**[*], **Margarita Chikobava**, **Ralf Romeike**, **Tim Landgraf**, and **Christoph Benzmüller**

Freie Universität Berlin, Germany
{veronika.solopova, oana-iuliana.popescu, margarita.chikobava, ralf.romeike, tim.landgraf, c.benzmueller}@fu-berlin.de

Abstract

Reflection about a learning process is beneficial to students in higher education (Bubnys, 2019). The importance of machine understanding of reflective texts grows as applications supporting students become more widespread. Nevertheless, due to the sensitive content, there is no public corpus available yet for the classification of text reflectiveness. We provide the first open-access corpus of reflective student essays in German. We collected essays from three different disciplines (Software Development, Ethics of Artificial Intelligence and Teacher Training). We annotated the corpus at sentence level with binary reflective/non-reflective labels, using an iterative annotation process with linguistic and didactic specialists, mapping the reflective components found in the data to existing schemes and complementing them. We propose and evaluate linguistic features of reflectiveness and analyse their distribution within the resulted sentences according to their labels. Our contribution constitutes the first open-access corpus to help the community towards a unified approach for reflection detection.

1 Introduction

Consciously experienced and reflected practice is a prerequisite for professionalization (Donald, 1983). For pre-service teachers, reflection is crucial because it belongs to the core competencies of prospective teachers (Combe and Kolbe, 2004; Hänsel, 1996; Shandomo, 2010). In literature, several types of reflection can be found. Core reflection deals with the core of one's personality: mission and identity (Korthagen and Vasalos, 2005), while self-reflection refers to thinking about one's own behaviour, actions, thoughts or attitudes (Bubnys, 2019). The reflection process can be either guided using prompts to indicate the structure of the reflection (Allas et al., 2020), or free, where the reflection process follows no given structure (Sturgill and Motley, 2014). In our corpus, we mainly focus on guided self-reflection.

[*] indicates equal contribution.

Educational staff must assess students' reflection texts, yet this is a non-trivial and time-consuming task. Machine learning methods can provide possibilities to create such applications. However, the first step towards this is identifying whether reflection is present in a text or not. Collections of student essays in machine-readable formats have been created for the last two decades for various machine learning tasks, such as automated essay scoring (Foltz et al., 1999), argumentation mining (Wang et al., 2020), reflection detection and automated feedback (Wulff et al., 2020). However, to the best of our knowledge, there is no open-source corpus of reflective essays currently available. The reason, in our opinion, lies in the challenges that this kind of data brings. From an ethical point of view, these data are sensitive, since they can be highly personal. In addition, essays are usually collected in an educational setting, and it might be against regulations to publish them. Furthermore, inspiring students to reflect is difficult. As a literature review shows, students mostly write descriptive sentences when journaling (Dyment and O'connell, 2010).

We thus contribute a publicly available, balanced text corpus of reflective and descriptive sentences from students of various universities and disciplines as the first step towards a benchmark for reflection detection in texts. For this, we collected essays from three different sources and anonymized them. We then pre-processed texts into sentences and added manual sentence level annotations according to a synthesised taxonomy, engaging professional linguists and didactic specialists to refine our criteria. We present our quantitative and qualitative linguistic analysis of the resulted corpus. The link to our data can be found in Appendix A.

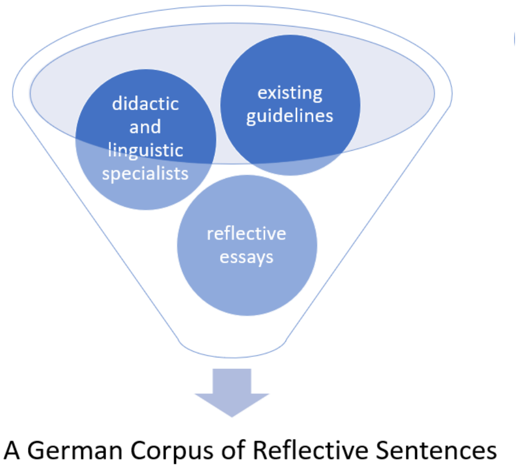

didactic and linguistic specialists
existing guidelines
reflective essays
A German Corpus of Reflective Sentences

Figure 1: The main components of our approach.

2 Related Work

In the context of the multi-genre essay collection, significant works include the British Academic Written English (BAWE) (Nesi and Gardner, 2012, 2013), the Uppsala Student English Corpus (USE) (Axelsson and Berglund, 2002), and the Michigan Corpus of Upper-Level Student Papers (MICUSP) (Römer and Swales, 2010). Several efforts were undertaken to create a specialized reflective corpus of students essays at sentence level, namely in pre-service and early teachers settings (Wulff et al., 2020; Murphy, 2015) or medical students and personnel (Liu et al., 2019a; Olex et al., 2020). For the didactic case specifically, there has been increasing work in automated detection of reflective sentences in the didactic context (Geden et al., 2021; Jung and Wise, 2020; Liu et al., 2019c; Wulff et al., 2020; Ullmann, 2019, 2017, 2015). However, none of the used corpora are publicly available.

3 Data Collection

We collected essays of different lengths in both English and German from students and pre-service teachers. We used the sentence segmenter of SpaCy (Honnibal et al., 2020) to obtain a total of 4232 sentences. During the annotation process, we performed manual anonymization, eliminated all the occurring personal information, including mentioned social media accounts, as well as student and teaching staff names. We describe below how data from the individual sources were collected. For more details on the segmentation, anonymization, and consent processes, see Appendix A.

Dundee teaching placement essays With the agreement of the University of Dundee, we scraped 122 reflective essays in English written by students in teacher training during their placements in primary and secondary school in 2018. The students had to upload their essays in the form of an e-Portfolio on Glow Blogs[1], a provider of WordPress tools used by the Scottish educational centers. The data reflect their impressions of the Scottish educational system in general and school approaches in particular, the acquired skills, their background, role models, insecurities, and motivations to become a teacher.

We translated the essays into German using DeepL[2] and manually corrected conflicting translations that occurred due to inconsistent formatting. After segmentation into simple sentences, we obtained a total of 3595 sentences.

Ethics of AI and Software development Using a questionnaire, we collected a set of guided reflective essays in German and English from students of the Free University and the Technical University of Berlin taking a Software Development project or the Ethics of AI lecture. Data was collected repeatedly at an interval of a few weeks.

The students were asked to reflect on the learning outcome since the previous collection. They were guided by a set of questions developed using Gibbs' reflective cycle (Gibbs, 1988), thus spanning the following topics: description of the action they took during their work/learning process, evaluating what they have learned and how to apply it further, what challenges they encountered, and which feelings they note. Additionally, they had to rate how their perception and their competencies of the topic changed and to describe why. After segmentation, we obtained a total of 637 sentences.

4 Annotation Guidelines

4.1 Reflection on the topic

Reflection on the topic accompanies the complex learning process and helps to integrate new knowledge into the existing one and further elaborate on it. In contrast to self-reflection, the object of reflection is part of the subject domain.

We developed our annotation criteria based on the Structure of the Observed Learning Outcome (SOLO) taxonomy (Biggs and Collis, 1982), which

[1]https://blogs.glowscotland.org.uk/glowblogs/
[2]https://www.deepl.com/translator

was proposed to assess the quality of learning. This taxonomy allows us to identify successful criteria, as it clearly defines the reflection steps. We adapted the three last levels of the taxonomy: multistructural, relational and extended abstract level. At the multistructural level, learners understand the relationship between different aspects but it's relationship to the whole remains unclear. At the relational level, aspects of knowledge are combined to form a structure. At the extended abstract level, knowledge is generalized to build a new domain. From the multi-structural level, we adapted the 'combine' action to the following criteria: (1) putting entities into relation (e.g., part of, opposite, but not providing an example). From the relational level, we adopted several criteria: (2) criticism, (3) evaluation and comparison between methods or objects, (4) analysis (e.g., causality, purpose, contributions), (5) classification and assessment of entities. Based on the last extended abstract level, we developed the two following criteria: (6) generating and formulating hypotheses and theorizing, (7) proposal of alternative implementation (suggestions how something could have been done in a different way).

4.2 Self-reflection

To annotate self-reflection, we adapted the schemes proposed by Shum et al. (2017) and Ullmann (2017), searching for evidence of the categories proposed by the authors in our own data. If the sentence met one or more of these requirements, we annotated it as reflective.

From Ullmann (2017) we included: (1) emotions and feelings, if they were followed by the cause or description of the circumstances which provoked them; (2) strategy adaptation based on previous experience, (3) different perspectives, and (4) outcome (lessons learned, future intentions, and action plans). From Shum et al. (2017), we implemented rhetoric components and expressions denoting: (5) learning something specific, (6) experimentation and ability, (7) increased confidence or ability, (8) applying theory into practice, (9) retrospection (e.g., *'it would have helped us'*, *'I should have done it'*), (10) expressions of reflecting specifically and (11) shifts in perception and beliefs. From the intersection of both schemes we included (12) personal beliefs, assumptions, self-assessment and (13) recognition of difficulties, which we aligned with rhetoric expressions of challenge and expressions describing the unexpected to prior assumptions.

We also introduced new categories based on our data and the didactic nature of our project: (14) rhetoric questions, (15) decisions (motives and the decision-making process), (16) motivation. We also determined conditional categories, that, similar to feelings, are annotated, taking into consideration the broader context and given reasons. These are opinions, evaluations, rendition of the words of others, generalisations, doubts (e.g., 'it seems', 'it may be'), 'even if A, not B' patterns, own interpretations of definitions, recommendations.

Contrary to Ullmann (2017) and Shum et al. (2017), we categorize descriptive sentences that describe the context of the event that triggers reflection as non-reflective. We support this decision by contrasting their linguistic feature distributions in Section 6.

5 Annotation Process

We manually annotated the collected sentences according to the synthesised guidelines presented in Section 4. If a sentence met at least one of the enumerated criteria we annotated it as reflective, even if it was a long sentence which also consisted of non-reflective components. The sub-corpora from the Software Project and Ethics of AI lectures were annotated in parallel by four annotators (the first authors and our two collaborating didactic specialists from the Friedrich-Alexander University Erlangen-Nürnberg). The initial inter-annotator agreement was low: 0.64 between first authors, 0.32 between first authors and didactic specialists, and 0.33 between the didactic specialists. Consequently, we refined our annotation guidelines and re-annotated the dataset. The Dundee sub-corpus was annotated by the first author, while the third author annotated 100 random sentences in order to verify consistency. The inter-rater agreement between the annotators was 0.66, which is considered substantial (Landis and Koch, 1977; Stemler and Tsai, 2008). Overall, we see that the annotation of reflectiveness is a problematic and tedious task, rather impossible using crowd-sourcing and requiring rounds of discussions and criteria harmonization among inter-disciplinary professionals, as also addressed by Ullmann (2019).

6 Analysis

6.1 Methodology

We investigate morphological features inspired from (Ullmann, 2015; Liu et al., 2019b; Murphy, 2015). However, we hypothesize that reflective sentences also differ in syntactic categories. Using a list of respective subordinate conjunctions and punctuation, we extracted main types of subordinate clauses and their length, e.g clause of purpose ('Within the framework of our group, we additionally met online on average once a week *to share research results and plan the next project steps.*', len=10); clause of reason ('I volunteered *because I want to learn to make better slides and I want to get better at presenting.*', len=17).

We compared the feature distribution in reflective versus non-reflective sentences. The resulted distribution of classes is balanced, with 2177 reflective and 1970 non-reflective sentences. We normalized feature counts according to the number of tokens per sentence, transforming them into frequency counts. As our features were mostly non-normally distributed, we applied non-parametric U-tests (Wilcoxon-Mann-Whitney) and multiple-test correction with Benjamini-Hochberg Procedure (N=45 tests). Since we find a large number of significant features, we further restricted our criteria. We filtered out features with medians lying on 0 (i.e., where more than 50% of the counts are 0), which is not taken into consideration by the U-test. Instead, it considers mean ranks, i.e., the arithmetic average of the positions in the list.

6.2 Results

The number of tokens in the sentence appears to be one of the most discriminating factors: reflective sentences tend to be longer, while non-reflective sentences are often nominal and/or contain short enumerations. At the same time, reflective sentences tend to be complex (with both subordinate or coordinate clauses using respective conjunctions). Relative clauses are the most frequent in reflective sentences, as they bring additional details describing the subject. Contrary to our expectations, the clauses of reason and purpose, typically used in justifications, show only a slight positive trend for reflective sentences in the Dundee sub-corpus, possibly because it often illustrates a situation and can contain descriptive causes and goals, e.g., *'We did not go outside because of the rain'*. The trend does become stronger in the self-reflection sub-corpora.

We can observe the presence of solid justification with our 'claims' feature, which checks matches with opinion words (e.g., 'standpoint', 'sure', 'convinced', 'opinion'), and 'supports', which is a collective count of subordinate clauses of reason, purpose, concession, condition and adversation. All subordinate clauses we measured are generally more present in the reflective part of the data set, and the mean length of clauses of reason and purpose is also generally longer. Concessive clauses appear to be the most numerous in this kind of texts. Reflective sentences also show higher probability of explicit coherency markers with discursive connectives (e.g., 'although', 'however').

As for the tenses used, reflective sentences are more often written using the Future tenses, while non-reflective utterances show slight preference of the Past tenses.

Our 'personalizing' marker, which shows usage of first person singular and plural of pronouns (personal, possessive and reflexive), is found to be significantly more present in reflective sentences, as also found by (Ullmann, 2015), as well as a number of adverbs, verbs and adjectives (Murphy, 2015). However, we also measured usage of the German indefinite impersonal pronoun *'man'*, which similarly to English pronoun 'one' can be considered a tool to generalize, distance the authors from the opinion they express, and make it less personal (hence,'distancing' feature). Counter-intuitively, it was also found slightly more used in reflective sentences, rather than in descriptive ones.

Interestingly, our data also shows a negative trend for lexical words in reflective sentences and a positive one for stop words, which means that reflective sentences tend to be wordier, but less informative.

High modality words (e.g. 'actually', 'categorically') strongly correlate with sentence reflectiveness, while modal verbs and subjunctive mood (German *Konjunktiv* I and II) show the same trend in all but Dundee sub-corpus. This trend discrepancies between the original German and translated English data calls for further investigation into differences between reflection articulation in different languages.

7 Conclusion

With the proposed corpus, we aim to make the first step towards a more unified approach to reflection detection. At the moment, it is not possible to compare existing models, as there is no publicly available benchmark for this task. To address this issue, we created an open-source annotated text corpus of reflective and descriptive sentences from students of various universities and disciplines. We also provide the quantitative and qualitative analysis of the gathered data and describe the annotation procedure and quality assurance measures we took.

Our work has several limitations. Our annotators are not native German speakers, which can influence labeling. However, this will be re-visioned with later versions of the corpus, as we plan to increase the number of annotators and include native speakers. Another drawback is the automatic translation of the English data into German. While we plan to quantitatively increase our corpus with German data in the future, the Dundee sub-corpus provides a valuable addition. This way, however, it largely influence the results for language-specific features such as subjunctive mood presence, which can appear in translations, but which are still much more common to German than to modern English.

We address the low inter-annotators agreement problem with harmonization sessions and refinement of the coding scheme to ensure coverage of complicated instances. We report that with each iteration, inter-annotator agreement increased significantly. Thus, we reckon that a fruitful discussion of linguists and specialists of the field in the focus of the task, being a time-consuming process, is the only probable answer to the annotation of cognitive, subjective categories.

8 Future work

Sentence level segmentation has significant disadvantage compared to text level processing. Nevertheless, for modern classification algorithms, there is a need for an immense amount of data points. Thus, we decide to trade off context for the sake of robustness. In the future, we aim to prove the hypothesis that textual level reflection can still be reconstituted, computing an overall reflectiveness score. Finally, binary classification is only the first step, while we plan to add a more granular reflection level categories according to (Fleck and Fitzpatrick, 2010), sentiment polarity, emotions and the position of the sentence in Gibb's cycle (Gibbs, 1988). We also plan to expand the corpus with a larger number of guided reflections from different disciplines. Our overall goal is automated reflective essay analysis, which we plan to compare to the existing results by (Ullmann, 2019; Wulff et al., 2020), in order to propose an adequate level of feedback that matches the student's needs.

9 Acknowledgements

We would like to thank our collaborators from Friedrich-Alexander University Erlangen-Nürnberg, and namely Chengming Zhang and Jessica Schiessl for their participation in the co-annotation process, as well as Prof. Dr. Michaela Gläser-Zikuda and Prof. Dr. Florian Hofmann for the sanity check of our guidelines from the pedagogic perspective. We are also very grateful to Derek Robertson and Dundee University for granting us permission to process and publish their Glow Blogs essays.

References

Raili Allas, Äli Leijen, and Auli Toom. 2020. Guided reflection procedure as a method to facilitate student teachers' perception of their teaching to support the construction of practical knowledge. *Teachers and Teaching*, 26(2):166–192.

Margareta Axelsson and Ylva Berglund. 2002. The uppsala student english corpus (use): a multi-faceted resource for research and course development. *Language and Computers*, pages 79–90.

John B. Biggs and Kevin F. Collis. 1982. The psychological structure of creative writing. *Australian Journal of Education*, 26:59 – 70.

Remigijus Bubnys. 2019. A journey of self-reflection in students' perception of practice and roles in the profession. *Sustainability*, 11:194.

Arno Combe and Fritz-Ulrich Kolbe. 2004. Lehrerprofessionalität: Wissen, können, handeln. In *Handbuch der Schulforschung*, pages 833–851. Springer.

A Donald. 1983. *The reflective practitioner: How professionals think in action*. Basic books.

Janet E. Dyment and Timothy S. O'connell. 2010. The quality of reflection in student journals: A review of limiting and enabling factors. *Innovative Higher Education*, 35:233–244.

Rowanne Fleck and Geraldine Fitzpatrick. 2010. Reflecting on reflection: Framing a design landscape. pages 216–223.

Peter Foltz, Darrell Laham, and T Landauer. 1999. Automated essay scoring: Applications to educational technology. *World Conference on Educational Multimedia, Hypermedia and Telecommunications*, 1.

Michael Geden, Andrew Emerson, Daniel Carpenter, Jonathan P. Rowe, Roger Azevedo, and James C. Lester. 2021. Predictive student modeling in game-based learning environments with word embedding representations of reflection. *Int. J. Artif. Intell. Educ.*, 31:1–23.

Graham R. Gibbs. 1988. Learning by doing: A guide to teaching and learning methods. Further Education Unit.

Dagmar Hänsel. 1996. *Lehrerbildung neu denken und gestalten*. Beltz.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Yeonji Jung and Alyssa Friend Wise. 2020. How and how well do students reflect?: multi-dimensional automated reflection assessment in health professions education. *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge*.

Fred Korthagen and Angelo Vasalos. 2005. Levels in reflection: Core reflection as a means to enhance professional growth. *Teachers and teaching*, 11(1):47–71.

J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33 1:159–74.

Ming Liu, Simon Buckingham Shum, Efi Mantzourani, and Cherie Lucas. 2019a. *Evaluating Machine Learning Approaches to Classify Pharmacy Students' Reflective Statements*, pages 220–230.

Ming Liu, Simon Buckingham Shum, Efi Mantzourani, and Cherie Lucas. 2019b. *Evaluating Machine Learning Approaches to Classify Pharmacy Students' Reflective Statements*, pages 220–230.

Ming Liu, Simon Buckingham Shum, Efi Mantzourani, and Cherie Lucas. 2019c. Evaluating machine learning approaches to classify pharmacy students' reflective statements. In *AIED*.

Bróna Murphy. 2015. A corpus-based investigation of critical reflective practice and context in early career teacher settings. *Classroom Discourse*, 6:107 – 123.

Hilary Nesi and Sheena Gardner. 2012. *Genres Across the Disciplines: Student Writing in Higher Education*.

Hilary Nesi and Sheena Gardner. 2013. A classification of genre families in university student writing. *Applied Linguistics*, 34:25–52.

Amy Olex, Deborah DiazGranados, Bridget Mcinnes, and Stephanie Goldberg. 2020. Local topic mining for reflective medical writing. *AMIA Joint Summits on Translational Science proceedings. AMIA Joint Summits on Translational Science*, 2020:459–468.

Ute Römer and John Swales. 2010. The michigan corpus of upper-level student papers (micusp). *Journal of English for Academic Purposes*, 9:249–249.

Hibajene M. Shandomo. 2010. The role of critical reflection in teacher education. volume 4, pages 101–113. School-University Partnerships.

S. B. Shum, Á. Sándor, R. Goldsmith, Randall Bass, and M. McWilliams. 2017. Towards reflective writing analytics: Rationale, methodology and preliminary results. *Journal of learning Analytics*, 4:58–84.

Steven E. Stemler and Jessica W Tsai. 2008. Best practices in interrater reliability three common approaches. In *Best Practices in Quantitative Methods*.

Amanda Sturgill and Phillip Motley. 2014. Methods of reflection about service learning: Guided vs. free, dialogic vs. expressive, and public vs. private. *Teaching and Learning Inquiry: The ISSOTL Journal*, 2:81–93.

T. Ullmann. 2015. Keywords of written reflection - a comparison between reflective and descriptive datasets. In *ARTEL@EC-TEL*.

T. Ullmann. 2017. Reflective writing analytics: empirically determined keywords of written reflection. *Proceedings of the Seventh International Learning Analytics & Knowledge Conference*.

Thomas Ullmann. 2019. Automated analysis of reflection in writing: Validating machine learning approaches. *International Journal of Artificial Intelligence in Education*, 29.

Hao Wang, Zhen Huang, Yong Dou, and Yu Hong. 2020. Argumentation mining on essays at multi scales. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5480–5493, Barcelona, Spain (Online). International Committee on Computational Linguistics.

P. Wulff, David Buschhüter, Andrea Westphal, Ann I. Nowak, Lisa Becker, Hugo Robalino, Manfred Stede, and Andreas Borowski. 2020. Computer-based classification of preservice physics teachers' written reflections. *Journal of Science Education and Technology*, 30:1–15.

A Data collection

The entire questionnaire, including the consent form, the code for linguistic feature annotation and the data-set divided into training and test sets for the benchmark purposes are available on OSF depository: https://osf.io/ug9r8/ and Github : https://github.com/oanaucs/german_reflective_corpus.

A.1 Guided Reflection Questions (German)

1. Bitte denken Sie an die Erfahrung die Sie während der Aufgabenlösung gemacht haben - aus Ihrer Perspektive. Wer war dabei, was haben Sie gelöst, wann und wo? Erklären Sie bitte welche Entscheidungen und warum Sie sie getroffen haben. Bitte schreiben Sie vollständige Sätze.

2. Bitte reflektieren Sie über das Gelernte durch die Aufgabenlösung. Was haben Sie gelernt? Sind Sie selbstbewusster geworden? Werden Sie das Gelernte in der Praxis anwenden? Was haben Sie vor? Was hätten Sie besser machen können? Bitte schreiben Sie vollständige Sätze.

3. Bitte denken Sie jetzt an die Schwierigkeiten die während der Aufgabenlösung aufgetaucht sind. Was waren die Herausforderungen? Ist etwas unerwartetes passiert? Haben Ihre vorherige Annahmen (z.B. Zeit für die Aufgabe) doch nicht gestimmt? Bitte schreiben Sie vollständige Sätze.

4. Erklären Sie bitte wie Ihre Wahrnehmung gegenüber das Thema verändert hat. Bitte schreiben Sie vollständige Sätze.

5. Erklären Sie bitte wie Ihre Wahrnehmung gegenüber Ihre Kompetenzen verändert hat. Bitte schreiben Sie vollständige Sätze.

6. Erklären Sie bitte wie sich während und nach der Aufgabenlösung gefühlt haben. Welche Emotionen haben Sie erlebt? Wie haben sich Ihre persönliche Überzeugungen verändert? Bitte schreiben Sie vollständige Sätze.

A.2 Guided Reflection Questions (English)

1. Please think about the experience you had while solving the task - from your perspective. Who was there, what did you solve, when and where? Please explain your decisions and why you made them. Please write complete sentences.

2. Please reflect on what you have learned through the assignment. What was new? Have you become more confident? Will you apply what you have learned in practice? What do you plan to do? What could you have done better? Please write complete sentences.

3. Please think now about the difficulties that arose during the task solution. What were the challenges? Did something unexpected happen? Were your previous assumptions (e.g., time for the task) not correct after all? Please write complete sentences.

4. Please explain how your perception towards the subject has changed. Please write complete sentences.

5. Please explain how your perception towards your competencies has changed. Please write complete sentences.

6. Please explain how you felt during the task and after solving it. What emotions did you experience? How did your personal beliefs change? Please write complete sentences.

Table 1: Linguistic features. The coloured features are the most relevant ones according to our analysis.

Feature	Effect size	P-value
Surface statistics		
Number of tokens	1389296.0	<0.001
Number of characters	2206504.0	0.273905
Stop words	1519336.0	<0.001
Lexical words	1584599.5	<0.001
Foreign words	2098738.5	<0.001
Negations	2042086.0	<0.001
Parts of Speech		
Number of adjectives	1945153.5	<0.001
Number of adverbs	1946744.0	<0.001
Number of prepositions	2188743.0	0.1456593
Number of demonstratives	2020975.0	<0.001
Number of numerals	2133089.5	1E-07
Number of proper nouns	2041038.0	<0.001
Number of nouns	1823595.5	<0.001
Number of pronouns	1677706.0	<0.001
Number of verbs	1754766.0	<0.001
Subordinate clauses		
Purpose	2133162.5	1.3e-06
Length of purpose	2143140.0	8.6e-06
Reason	1961005.5	<0.001
Length of reason	2084323.0	<0.001
Condition	2080748.5	<0.001
Consecutive	2225781.0	0.2218441
Temporal	2194652.5	0.0291472
Modal	2200269.5	0.0009057
Relative	2069188.5	3E-07
Consession	2187280.5	1.28e-05
Adversation	2226559.0	0.2559304
General Syntax		
Coordination conjunctions	1825060.5	<0.001
Subordination conjunctions	1631024.0	<0.001
Complex sentences	1644350.0	<0.001
Simple sentences	1868503.5	<0.001
Moods		
Modal verbs	2134704.0	<0.001
Subjunctive	1995376.5	<0.001
High modality words	1906942.0	<0.001
Patterns		
I+ finite verb	2006681.5	<0.001
To be + adjective	1968976.5	<0.001
Justification words		
Claims	1909487.0	<0.001
Supports	1800356.5	<0.001
Miscellaneous		
Discourse markers	1952871.0	<0.001
Personalizing	1687035.5	<0.001
Distansing	2181423.0	0.0001956
Tenses		
Present	2108015.5	0.0003568
Future	2175966.0	1.06e-05
Past	2138633.0	0.0051358

Analysis of Manipuri Tones in ManiTo: A Tonal Contrast Database

Thiyam Susma Devi
IIT Guwahati
India
thiyam.devi@iitg.ac.in

Pradip K. Das
IIT Guwahati
India
pkdas@iitg.ac.in

Abstract

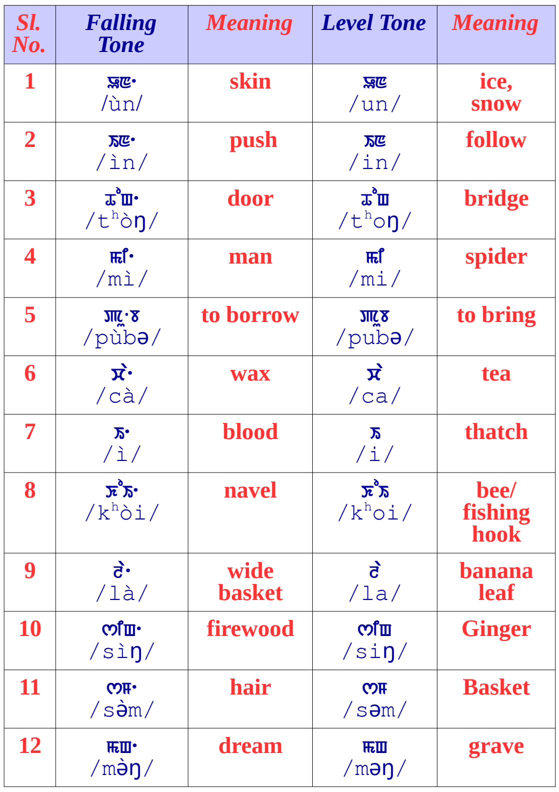
Manipuri is a low-resource, tonal language spoken mainly in India's northeastern state, Manipur. It has two tones - level and falling tones. For an acceptable Automatic Speech Recognition (ASR) system, integrating tonal cues from a potent Tone Recognition model is essential. ASR research on tonal languages, African, Asian and Indo-European tonal languages such as Thai, Chinese, Vietnamese and Mandarin have been done, but Manipuri is underexplored. This paper focuses on the fundamental analysis of the developed hand-crafted tonal contrast dataset, ManiTo. It is observed that the height and slope of the pitch contour can be used to distinguish the two tones of the Manipuri language.

1 Introduction

Automatic Speech Recognition (ASR) generates the transcript of a spoken speech. It plays a significant role in stopping the extinction of endangered languages and is also required for fostering economic growth and benefits. For improving the performance of ASR, tone recognition carries a crucial role in tonal languages where tone distinguishes the meaning of the words. Intensive work has been done on tonal languages like Mandarin, Thai, Vietnamese, Chinese, etc., (Kaur et al., 2020), but no work has been done on the ASR of Manipuri.

Manipuri is one of the Indian Tibeto-Burman languages spoken in Manipur, a northeastern state of India. It is a tonal language in which the tone differentiates the word's meaning. So, robust tone recognition can enhance the speech-understanding tasks of Manipuri. It has its script known as Meitei Mayek (Singh et al., 2007). The script has twenty-seven main alphabets (Mapung Mayek), eight unreleased characters (Lonsum Mayek), eight vowel signs (Cheitap Mayek), three punctuation marks, including diacritics or tone marker (Khudam Mayek) and Cheising Mayek for the numerals.

Manipuri has two tones (Thoudam, 1980; Khan, 1987; Chelliah, 1992; Devi, 2004; Singh et al., 2007), level and falling tones. Each one of the syllables in Manipuri bears one of the two tones. The level tone is unmarked, while the falling tone is marked with /̀/ in English representation and with the falling tone marker or lum mayek, "꯬" just after the syllable in the Manipuri script. Some of the tonal contrast words pair with their meanings are shown in Figure 1.



Sl. No.	Falling Tone	Meaning	Level Tone	Meaning
1	ꯎꯟ꯬ /ùn/	skin	ꯎꯟ /un/	ice, snow
2	ꯏꯟ꯬ /ìn/	push	ꯏꯟ /in/	follow
3	ꯊꯣꯡ꯬ /tʰòŋ/	door	ꯊꯣꯡ /tʰoŋ/	bridge
4	ꯃꯤ꯬ /mì/	man	ꯃꯤ /mi/	spider
5	ꯄꯨ꯬ꯕ /pùbə/	to borrow	ꯄꯨꯕ /pubə/	to bring
6	ꯆꯥ꯬ /cà/	wax	ꯆꯥ /ca/	tea
7	ꯏ꯬ /ì/	blood	ꯏ /i/	thatch
8	ꯈꯣꯏ꯬ /kʰòi/	navel	ꯈꯣꯏ /kʰoi/	bee/ fishing hook
9	ꯂꯥ꯬ /là/	wide basket	ꯂꯥ /la/	banana leaf
10	ꯁꯤꯡ꯬ /sìŋ/	firewood	ꯁꯤꯡ /siŋ/	Ginger
11	ꯁꯝ꯬ /sə̀m/	hair	ꯁꯝ /səm/	Basket
12	ꯃꯡ꯬ /mə̀ŋ/	dream	ꯃꯡ /məŋ/	grave

Figure 1: List of tonal contrast word pairs with their meaning.

2 Related Works

Internationally, progressive work on tone recognition has been done for tonal languages like Mandarin, Thai, Cantonese, and Vietnamese for the last three decades. (Linkai et al., 2021) recognized lexical tones in Mandarin speech on the 863 corpora using a model that integrates multiple features at a different scale. Experimental results showed that their technique achieved a Tone Error Rate of 10.5%. (Nguyen et al., 2016) proposed an acoustic model using the tone feature to investigate the effect of tone in the Vietnamese Large Vocabulary Continuous Speech. The result improved the perception of phonemes by 19.25% against the non-tonal phonemes system. (Krittakom and Narissara, 2015) analyzed Neuro-fuzzy based approach to recognize Thai speech. Their dataset consisted of eight Thai words recorded in different environments. The result showed that the system was robust to noise and could yield higher recognition than other recognizers. In India, some work has been done on the tonal languages Punjabi and Mizo. (Gogoi et al., 2020) proposed a technique for Mizo tone recognition using Support Vector Machine (SVM) and Deep Neural Network (DNN) based classifiers. The experiment is performed on a dataset obtained from nineteen speakers. It achieved an accuracy of 73.39% for the SVM model and 74.11% for the DNN model. (Jyoti and Achyuta, 2020) proposed an ASR system based on pitch-dependent features and the probability of voicing estimated features. The results significantly improved the word error rate of the system.
However, there is no work done on the ASR of the Manipuri Language, which motivates us to build a dataset that contains the tonal contrast word pairs of Manipuri and examine the tone information to develop a robust ASR system for Manipuri.

3 Manipuri Tonal Contrast Dataset

A Manipuri tonal contrast dataset, ManiTo consisting of 3000 speech samples collected from

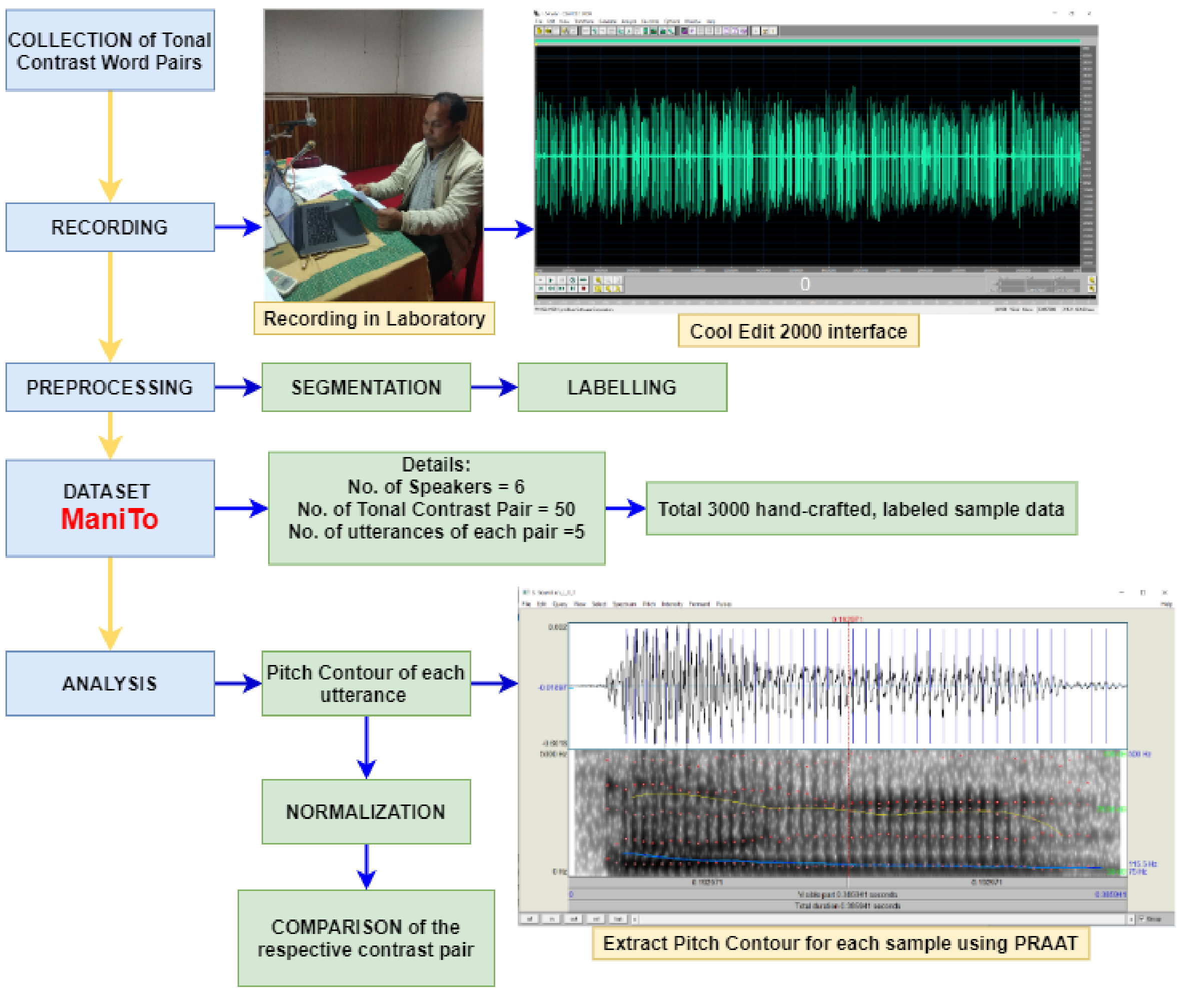

COLLECTION of Tonal Contrast Word Pairs
RECORDING
Recording in Laboratory
Cool Edit 2000 interface
PREPROCESSING
SEGMENTATION
LABELLING
DATASET ManiTo
Details:
No. of Speakers = 6
No. of Tonal Contrast Pair = 50
No. of utterances of each pair =5
Total 3000 hand-crafted, labeled sample data
ANALYSIS
Pitch Contour of each utterance
NORMALIZATION
COMPARISON of the respective contrast pair
Extract Pitch Contour for each sample using PRAAT

Figure 2: Flowchart depicting the summary of the work done

6 native speakers, 3 males and 3 females aged 20-35, is developed. The upper portion of the flowchart in Figure 2 shows the steps for the development of ManiTo. The corpus contains a total of 50 tonal contrast word pairs collected from different sources (Khan, 1987; Takhellambam, 2014; Thoudam, 1980; Chelliah, 1997; Singh, 2019), five utterances of each pair recorded using Cool Edit 2000 in a laboratory as well as in a quiet office environment. The recorded speech is further examined and manually segmented, keeping 1000 samples of silence at the prefix and suffix of the speech sample and saved as .wav format. Each file has the naming convention: WordName_ToneType_UtteranceID_SpeakerID. For example, un_f_2_3.wav means the particular speech sample is the falling tone"un" sound spoken by speaker 3, second utterance. The corpus ManiTo consists of hand-crafted labeled speech data of size 273MB.

4 Experimental Analysis

Praat, version 6.1.51 (Boersma and Van Heuven, 2001) tool is used to analyze speech data in ManiTo. The lower portion of the flowchart in Figure 2 shows the steps for analyzing the speech data.

4.1 Pitch Extraction

Feature extraction plays a vital role in developing a robust tone recognition system. As the pitch contour carries the salient information regarding tones, the fundamental frequency (F0) is generally used for tone recognition (Chao et al., 2019).
Initially, the pitch listing of a particular tonal contrast pair for a speaker, i.e., ten samples, 5 for falling and 5 for level, are retrieved using Praat. Praat is built with the most precise pitch extraction algorithm (Boersma and Van Heuven, 2001). All pitch values are collected and stored in different files.

4.2 Normalization

Normalization is performed on each extracted pitch listing value to compare the speech data efficiently with one another. For particular word pairs of a speaker, it contains 5 utterances of level tone speech and 5 utterances of falling tone speech. The normalization obtained the same length pitch listing values of the specific tone type utterances, i.e., after the normalization, the five utterances will have the same length of pitch listing values. Algorithm 1 shows the normalization method that we employed on the recorded data.

Algorithm 1: Normalization of a particular speech sample

```
Input:
$f0[L]$: Pitch listing containing L values
$L_{max}$: Maximum Pitch listing length
Output:
$Norm_{f0}[L_{max}]$: Pitch listing with $L_{max}$ values
begin
    Step 1: $k \leftarrow 0$                        ▷ $f0$ index
    Step 2: $j \leftarrow 0$           ▷ track insertion point
    Step 3: if $L < L_{max}$ then
        $insert_{Loc} \leftarrow L/\left(L_{max} - L\right)$
        for $i \leftarrow 0$ to $L_{max}$ do
            $Norm_{f0}[i] \leftarrow f0[k]$
            if $j = insert_{Loc}$ then
                $i \leftarrow i + 1$
                $Norm_{f0}[i] \leftarrow f0[k]$
                $j \leftarrow 0$
            end
            $k \leftarrow k + 1$
            $j \leftarrow j + 1$
        end
    end
end
```

4.3 Results

The analysis on the Tonal Contrast word pair_10, i.e., "sing" based on the pitch contour of each sample, is shown in Figure 3. The speech sounds of three male and two female speakers are analyzed separately as the female pitch is not comparable to the male pitch. The left column shows the analysis of male speech and the right for the female speech data. Figure 3a shows the normalized pitch contour of fifteen utterances of falling tone "sing" of the three male speakers, Figure 3c shows level tone normalized pitch contour, Figure 3e compares the two tones. Similarly, for female speech sounds, Figure 3b shows normalized pitch contour of ten utterances of falling tone "sing" and, Figure 3d shows the level tone and finally Figure 3f compares the two tones. From the plotted graph, we can initially claim that we can use the slope and height of pitch contour to distinguish the two tones, falling and level tones of Manipuri. The pitch of the level tone is lower than that of falling tone.

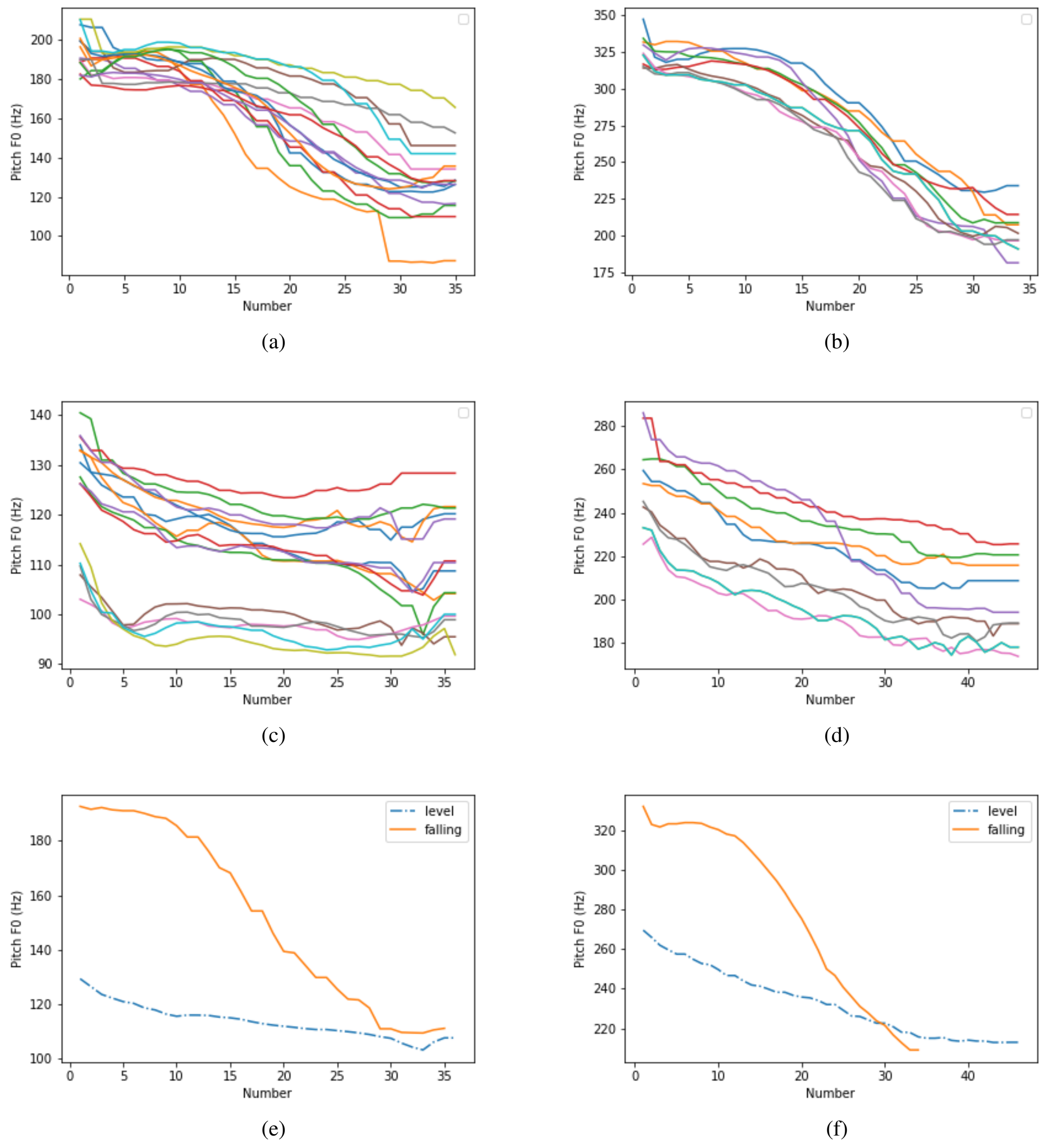


Figure 3: Normalized Pitch Contour of (a) 3 Male Speakers “sing” 15 utterances(falling tone) (b) 2 Female Speaker “sing” 10 utterances(falling tone) (c) 3 Male Speakers “sing” 15 utterances(level tone) (d) 2 Female Speaker “sing” 10 utterances(level tone) (e) Comparison of male average falling and level pitch (f) Comparison of female average falling and level pitch

5 Conclusion and Future Work

Initial analysis of the ManiTo, Manipuri Tonal Contrast dataset is performed. It is inferred that we can use the pitch contour to distinguish the two tones, falling and level tones. Further analysis is presently being done to precisely differentiate the tones and build a tone recognition system for Manipuri. Currently the dataset consists of 3000 samples from 6 speakers and unprocessed recordings from 3 more speakers. In the future, we will extend the dataset for 20 speakers and this will be made available for the speech community.

Acknowledgments

This work was supported by the Linguistic Department of Manipur University, Imphal. The authors would like to thank Prof. Chungkham Yashawanta Singh, Nameirakpam Amit, Laishram Niranjana and Dr. Yumnam Aboy Singh for their valuable advice.

References

Paul Boersma and Vincent Van Heuven. 2001. Speak and unSpeak with PRAAT. *Glot International*, 5.

Hao Chao, Cheng Song, Bao yun Lu, and Yong li Liu. 2019. Feature Extraction based on DBN-SVM for Tone Recognition. *Journal of Information Processing Systems*, 15(1):91–99.

Shobhana L. Chelliah. 1992. Tone in Manipuri. In *K. L. Adams and T. J. Hudak (Eds.), Papers from the first annual meeting of the Southeast Asian Linguistics Society. Tempe, AZ: Arizona State University.*

Shobhana L Chelliah. 1997. A grammar of Meitei. In *Mouton de Gruyter, Berlin, New York*.

Hajarimayum Subadani Devi. 2004. Loanwords in Manipuri and their impact. In *Linguistics of the Tibeto-Burman Area, Volume 27.1 Spring*.

Parismita Gogoi, Abhishek Dey, Wendy Lalhminghlui, Priyankoo Sarmah, and S. R. Mahadeva Prasanna. 2020. Lexical Tone Recognition in Mizo using Acoustic-Prosodic Features. In *Proceedings of The 12th Language Resources and Evaluation Conference, LREC 2020, Marseille, France, May 11-16, 2020*, pages 6458–6461. European Language Resources Association.

Guglani Jyoti and Mishra Achyuta. 2020. Automatic speech recognition system with pitch dependent features for Punjabi language on KALDI toolkit. *Applied Acoustics*, 167:107386.

Jaspreet Kaur, Amitoj Singh, and Virender Kadyan. 2020. Automatic Speech Recognition System for Tonal Languages: State-of-the-Art Survey. *Archives of Computational Methods in Engineering*, 28.

Abdul Ghaffar Khan. 1987. A Contrastive Study of Manipuri (Meiteilon) And English Phonology. In *Thesis of Doctor of Philosophy, Guwahati University*.

Srijiranon Krittakom and Eiamkanitchat Narissara. 2015. Thai speech recognition using Neuro-fuzzy system. In *2015 12th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, pages 1–6.

Peng Linkai, Dai Wang, Ke Dengfeng, and Zhang Jinsong. 2021. Multi-scale model for mandarin tone recognition. In *2021 12th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pages 1–5.

Quoc Bao Nguyen, Tat Thang Vub, and Chi Mai Luong. 2016. The Effect of Tone Modeling in Vietnamese LVCSR System. In *Procedia Computer Science*.

Chungkham Yashwanta Singh. 2019. Manipuri Grammar, 2nd Edition,. In *Textbook, Rajesh Publications*.

Leihaorambam Sarbajit Singh, Kabita Thaoroijam, and Pradip Kumar Das. 2007. Written Manipuri (Meiteiron) from Phoneme to Grapheme. In *Language in India, Volume 7:6*.

Meiraba Takhellambam. 2014. Tones in Meiteilol: A Phonetic Description. In *Language in India*.

Purna Chandra Thoudam. 1980. A Grammatical Sketch of Meiteiron. In *Thesis of Doctor of Philosophy, Jawaharlal Nehru University, New Delhi*.

Building a Linguistic Resource: A Word Frequency List for Sinhala

Aloka Fernando
University of Moratuwa
Sri Lanka
alokaf@uom.lk

Gihan Dias
University of Moratuwa
Sri Lanka
gihan@uom.lk

Abstract

A word frequency list is a list of unique words in a language along with their frequency count. It is generally sorted by frequency. Such a list is essential for many NLP tasks, including building language models, POS taggers, spelling checkers, word separation guides, etc., in addition to assisting language learners. Such lists are available for many languages, but a large-scale word list is still not available for Sinhala. We have developed a comprehensive list of words, together with their frequency and part-of-speech (POS), from a large textbase. Unlike many other such lists, our list includes a large number of low-frequency words (many of which are erroneous), which enables the analysis of such words, including the frequencies of errors. In addition to the main list, we have also prepared a list of linguistically verified words. The word frequency list and the verified word list are the largest collections of words lists that are available for the Sinhala language.

1 Introduction

Word frequency lists are useful for analysing the vocabulary of a language (Nation and Waring, 1997). They generally comprise a list of lexical words sorted by frequency of occurrence in a corpus, as a ranked list. Such lists can be used to build a directory, for language learning and for teaching. They are also useful in downstream applications such as part-of-speech (POS) tagging, syntactic parsing, machine translation, speech processing etc.

In this paper, first, we present a word frequency list for Sinhala. We compiled a list of 2 million unique words by considering published documents from multiple sources, representing diverse domains. The raw corpus contained 127 million word tokens. In addition to frequency, the list provides the POS Tag of each word. Second, we have compiled a linguistically verified word list of over 280 thousand unique words.

These two word lists are currently the largest available word lists for the Sinhala language. Therefore, they would be very useful linguistic resources for many downstream applications.

Further, both the word frequency list and verified word list are publicly released [1], for the progression of future research.

1.1 Objectives

In our work, we often encountered a word, and needed to know if it is frequently used, a rare word, or a misspelling. Therefore, we decided to take a large dataset including common-crawl data and other available data, and construct a list of words seen in the wild. Rather than a curated corpus, we wanted to use the largest dataset available. Instead of limiting ourselves to correctly spelled and separated words, we decided to build a list which includes incorrect words. As we often needed to know the POS of each word, we also tagged each word with a POS, even though the tagging may sometimes be inaccurate.

We also needed lists of correct words, therefore we decided to compile a list of linguistically verified words.

In this paper, Section 2 covers related work and Section 3 describes how the lists were compiled. Our results are analysed in Section 4, and Section 5 gives our conclusions and planned continuations.

2 Related Work

Word frequency lists have long been in use for languages such as Spanish, French, German and English. Before the advent of computing, they were generated manually, but have become much easier

[1] https://github.com/nlpcuom/Word-Frequency-List-for-Sinhala

to generate using a computer (Davies and Davies, 2017; Tschirner et al., 2019; Lonsdale and Le Bras, 2009; Leech et al., 2014).

The main objectives of many word frequency lists based on written or spoken corpora (Dang et al., 2017) were for language teaching and learning.

However, word lists are valuable resources for computational linguistics as well. They provide annotated datasets for downstream applications such as POS taggers (Kucera and Francis, 1967), dependency parsers (Silveira et al., 2014) etc.

More recently, frequency lists were generated for non-Latin languages such as Tamil (Kumar, 2019) and Russian (Sherstinova et al., 2020). Frequency lists were also generated for specific domains such as literary genres and historical periods, and even individual books and authors.

For Sinhala, a corpus based lexicon were done by Weerasinghe et al. (2009). This contains 35K unique words. They have constructed a word frequency list extracted from publicly available Sinhala text documents from multiple domains. However, the list does not appear to be publicly accessible.

More recently, Google has released a Sinhala lexicon with about 42,000 entries (Jansche, 2017).

Text documents were crawled from the web at scale (Schwenk et al., 2019; Wenzek et al., 2020) for the progress of research in diverse domains. Such crawled corpora are also available for Sinhala. To the best of our knowledge, the generated word frequency lists had not considered these recently web crawled data.

3 Methodology

3.1 Dataset for the Word Frequency List

As our dataset, we used publicly available Sinhala text crawled from the web by the Common Crawl project (Wenzek et al., 2020). We also included government documents (Fernando et al., 2020) and news sites (Isuranga et al., 2020). These documents contain text corresponding to modern usage of the language. Hence the word list would be suitable for many downstream applications.

The corpus statistics of the data is in Table 1.

Domain	Total Sents.	Total Tokens
Govt. Documents	77,694	0.75M
News	332,793	20M
Common crawl	5,000,324	106M

Table 1: Corpus Statistics

3.2 Data Cleaning and Pre-processing

The government documents dataset had been compiled manually and the text was of good quality. Since the news data and common crawl data had been crawled from the web, they needed to be cleaned before use.

The data were tokenised using the Sinhala tokenizer (Farhath et al., 2018), to separate punctuation and the words. As compound nouns, verbs or particles used as suffixes, may be written with or without white space, such words were not combined or split, but considered them as they appeared in the corpus.

The first cleaning step was to filter out words with non-Sinhala characters, including English and other foreign language words and numerals. Scripts were developed using rules and regular expressions to remove such invalid words.

The data contained Unicode errors such as duplicated modifiers, misplaced modifiers, separation of modifiers into parts, etc. These were corrected with the Unicode Error Corrector[2], a rule-based tool developed for Sinhala Unicode error correction. Some corrections were:

අාකර්ෂනය → ආකර්ෂනය

ෙඩාලර් → ඩොලර්

එ + ් → ඒ

නිරීක්්ෂණය → නිරීක්ෂණය

The Sinhala yansaya and rakaransaya symbols are formed using the Unicode zero-width joiner (ZWJ) character. Use of these symbols is mandatory in Sinhala. However, some systems erroneously delete ZWJ characters, giving incorrect words, e.g. ක්‍රිකට් (cricket) is erroneously depicted as ක්රිකට්. Others divide a word into two, e.g., ක්‍රිකට් → ක් රිකට් by replacing the ZWJ with a space.

These errors were corrected using our Zero-Width Joiner Fix [3], which uses lists of valid words and sub-words to identify where the ZWJ has been deleted or replaced, and re-inserts the character.

As there remained further invalid words in the list, they were removed based on a POS filtering. A Sinhala POS Tagger (Fernando and Ranathunga, 2018) was used to tag the words. We removed

[2]https://nlp-tools.uom.lk/uniec/

[3]https://nlp-tools.uom.lk/zwjfix/

the words tagged as Full Stop (FS), Punctuation (PUNC), Foreign Word (FRW) and Unknown (UNK). Subsequently, some of the obvious erroneous words were removed manually.

Sinhala contains many single character words and particles, such as ඒ (that) ද (and) ලී (wood). We included such words in our list as well.

3.3 Word Frequency list

After cleaning the final corpus statistics are shown in Table 2.

	Total Words
Tokens in original documents	127M
Total word count after cleaning	122,998,105
Total unique words	2,170,052

Table 2: Corpus statistics in-terms of token counts

We extracted the lemma of each word using SinMorphy (Kumarasinghe et al., 2021), a morphological parser for Sinhala. The top 20 frequent words are listed along with the lemma, POS and frequency in Table 3.

Words	Lemma	Frequency	POS Tag
මේ	මේ	1067105	DET
ඒ	ඒ	946049	ABB
ඇති	ඇති	567890	NIP
සහ	සහ	518043	CC
හා	හා	511663	CC
එක	එක	491217	NUM
මම	ම	485869	PRP
බව	බව	469031	POST
ද	ද	453787	RP
නම්	නම	438329	POST
කර	කර	423842	VNF
වන	වන	387674	VP
කරන	කර	376601	VP
අතර	අතර	353691	POST
මට	ම	342623	PRP
ගැන	ගැන	339464	POST
මෙම	මෙම	326924	DET
නෑ	නෑ	323017	NIP
නිසා	නිසා	306021	POST
වූ	ව	298454	VP

Table 3: Top 20 Frequent words

The most frequent words are mainly determiners, particles, pronouns, etc. However, we see that the POS tagging is sometimes not accurate.

3.4 Verified Words List

The words in the Word Frequency list were run through a spelling checker (Liyanapathirana et al., 2021), and the words accepted by it were taken as correct. As it is infeasible to manually check all the words which failed the spelling check, it was decided to manually check the 3555 highest frequency words which failed the spelling check. Of these, 1836 were manually verified to be correct, and added to the verified words list. This list comprises of 280,603 words. This is the largest list of verified Sinhala words available.

4 Analysis

4.1 Analysis of the Word Frequency List

When the distribution of words was analysed, we observed that 50% of the words in the corpus are covered by 17% of the words in the word frequency list. This means 17% words can be identified as the most commonly used words in Sinhala language.

Subsequently we analysed the word counts based on the POS Tag. The outcome is shown in Table 4.

Gram. Category	POS Categories	Total Words	Per. %
Noun	NNC,NNP, PRP,NNJ, VNN,NNP	67.8M	55.1
Verb	VNF,VP,VFM	18.7M	15.2
Adjective	JJ	9.0M	7.4
Adverb	RB	1.4M	1.2
Other		26.1M	21.2

Table 4: Word counts based on POS Tag.

Words tagged as Common Noun (NNC), Proper Noun(NNP), Pronoun(PRP), Adjectival Noun(NNJ) and Verbal Noun(VNN) were grouped into the Noun grammatical category. Those tagged as Verb Non Finite(VNF), Verb Finite(VFM) and Verb Participle(VP) were considered as Verbs.

From the statistics in Table 4, we see that the majority of words in the list are nouns. Sinhala is a morphologically rich language, which means the words are inflected based on the gender, number, case, etc. As nouns have more morphological variants, they account for more entries in the list.This emphasises the importance for linguistic tool support for nouns in morphologically rich languages such as Sinhala.

On the other hand number of verbs, adjectives and adverbs in Sinhala is limited. Therefore the representation of such words in the list is lower.

When we further analysed the words under each POS category, we came across some issues.

Some words were incorrectly classified by the POS tagger, eg: අකකර (acres) is a noun but was incorrectly tagged as a non-finite verb (VNF) . We plan to implement a better POS tagger at a later date.

Further, some words contained spelling and word separation issues, e.g. නිලධාරී → නිලධාරි (official), අංකගණිතය → අංක ගණිතය (arithmetic). These may or may not be considered errors depending on the point of view of the user.

4.2 Analysis of the Verified Word List

Each word in the linguistically verified word list was parsed by the morphological parser (Kumarasinghe et al., 2021) and the most frequent lemmas were analysed. The morphological parser is rule-based, and covers both morphological rules as well as sandhi-rules. Further these rules have been linguistically verified. Therefore we believe the morphological parser can provide reliable morphological information for our word list.
Of the 280,603 unique words in the linguistically verified word list, we obtained morphological information for 256,083 words, which is a coverage of 91%. A total of 43,313 unique lemmas were found. The information corresponding to the top most 10 frequent lemmas can be found in Table 5. The Total Words, in Table 5 corresponds to the total number of words from the word frequency list, with the lemma.
It was an interesting observation that 50% of total words in the word list were covered by only 10% of unique lemmas.

5 Conclusion and Future work

The word frequency list, comprising over 2 million words, is by far the largest word list for the Sinhala language. It is also the only one supplemented with POS information. Although the list contains many incorrect words and incorrect word separations, this is a feature, not a drawback, as it allows us to analyse the frequencies of variant spellings and compound words.

The verified word list of over 280 thousand words is also the largest such list for Sinhala. Although many words and word inflections are not in this list, it does cover most of the words in common use.

Lemma	Total Words ('000)	Sample Words
කර	1,674	කර, කරන, කරමින්...
ම	1,124	මම, මට, මා, මමත්, මටත්...
ඒ	1,072	ඒ, ඒත්, ඒයි, ඒය, ඒවල...
මේ	1,067	මේ, මේට, මේගේ, මේටත්...
ව	902	වූ, වන්නේ, වෙයි, නොවන...
අප	865	අපි, අපේ, අප, අපිට...
ය	738	ය, ගිය, ගිහින්, ගියා, යයි...
එක	723	එක, එකේ, එකෙන්, එකත්...
බව	649	බව, බවයි, බැවිනි, බැවිණී...
නම	625	නම්, නමුත්, නම, නමක්...

Table 5: Top 10 Most Frequent Lemmas

As future work, we plan to compile a frequency lists of morphosyntactic suffixes for Sinhala. We also plan to compile lists of word bi-grams, tri-grams, etc.

We also plan to compile domain-specific word lists for government documents, news, textbooks, web, etc.

These lists have been used to develop several other tools, including a spelling checker and a sentence generator, and are being used to identify spelling error patterns, etc. It will be a useful tool in many other areas of NLP.

6 Acknowledgments

This research was supported by the Accelerating Higher Education Expansion and Development (AHEAD) Operation of the Ministry of Education funded by the World Bank.

References

Thi Ngoc Yen Dang, Averil Coxhead, and Stuart Webb. 2017. The academic spoken word list. Language Learning, 67(4):959--997.

Mark Davies and Kathy Hayward Davies. 2017. A frequency dictionary of Spanish: Core vocabulary for learners. Routledge.

Fathima Farhath, Surangika Ranathunga, Sanath Jayasena, and Gihan Dias. 2018. Integration of bilingual lists for domain-specific statistical machine translation for sinhala-tamil. In 2018 Moratuwa

Engineering Research Conference (MERCon), pages 538--543. IEEE.

Aloka Fernando, Surangika Ranathunga, and Gihan Dias. 2020. Data augmentation and terminology integration for domain-specific sinhala-english-tamil statistical machine translation. arXiv preprint arXiv:2011.02821.

Sandareka Fernando and Surangika Ranathunga. 2018. Evaluation of different classifiers for sinhala pos tagging. In 2018 Moratuwa Engineering Research Conference (MERCon), pages 96--101. IEEE.

Udhan Isuranga, Janaka Sandaruwan, Udesh Athukorala, and Gihan Dias. 2020. Improved cross-lingual document similarity measurement. In 2020 International Conference on Asian Language Processing (IALP), pages 45--49. IEEE.

Martin Jansche. 2017. A pronunciation dictionary for sinhala.

H Kucera and W Nelson Francis. 1967. The brown university standard corpus of present day american english.

LR Prem Kumar. 2019. Word frequency in language teaching--a case study of tamil textbooks of tamilnadu. Language in India.

Kalindu Kumarasinghe, Gihan Dias, and Indu Herath. 2021. Sinmorphy: A morphological analyzer for the sinhala language. In 2021 Moratuwa Engineering Research Conference (MERCon), pages 681--686. IEEE.

Geoffrey Leech, Paul Rayson, et al. 2014. Word frequencies in written and spoken English: Based on the British National Corpus. Routledge.

Upuli Liyanapathirana, Kaumini Gunasinghe, and Gihan Dias. 2021. Sinspell: A comprehensive spelling checker for sinhala.

Deryle Lonsdale and Yvon Le Bras. 2009. A frequency dictionary of French: Core vocabulary for learners. Routledge.

Paul Nation and Robert Waring. 1997. Vocabulary size, text coverage and word lists. Vocabulary: Description, acquisition and pedagogy, 14:6--19.

Holger Schwenk, Guillaume Wenzek, Sergey Edunov, Edouard Grave, and Armand Joulin. 2019. Ccmatrix: Mining billions of high-quality parallel sentences on the web. arXiv preprint arXiv:1911.04944.

Tatiana Sherstinova, Alexander Grebennikov, Tatiana Skrebtsova, Anna Guseva, Mary Gukasian, Irina Egoshina, and Maria Turygina. 2020. Frequency word lists and their variability (the case of russian fiction in 1900-1930). In Conference of Open Innovations Association, FRUCT, 27, pages 366--373. FRUCT Oy.

Natalia Silveira, Timothy Dozat, Marie-Catherine De Marneffe, Samuel R Bowman, Miriam Connor, John Bauer, and Christopher D Manning. 2014. A gold standard dependency corpus for english. In LREC, pages 2897--2904. Citeseer.

Erwin Tschirner, Jupp Möhring, and Elisabeth Muntschick. 2019. A frequency dictionary of German: Core vocabulary for learners. Routledge.

Ruvan Weerasinghe, Dulip Herath, and Viraj Welgama. 2009. Corpus-based sinhala lexicon. In Proceedings of the 7th Workshop on Asian Language Resources (ALR7), pages 17--23.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Édouard Grave. 2020. Ccnet: Extracting high quality monolingual datasets from web crawl data. In Proceedings of the 12th Language Resources and Evaluation Conference, pages 4003--4012.

Part of Speech Tagging for a Resource Poor Language : Sindhi in Devanagari Script using HMM and CRF

Bharti Nathani
Department of computer Science
Banasthali Vidyapith
Banasthali
bhartinathani@rediffmail.com

Nisheeth Joshi
Department of Computer Science
Banasthali Vidyapith
Banasthali
nisheeth.joshi@rediffmail.com

Abstract

Part of speech tagging is a pre-processing step of various NLP applications. Mainly it is used in Machine Translation. This research proposes two POS taggers, i.e., an HMM-based and CRF based tagger. To develop this tagger, the corpus of manually annotated 30,000 sentences has been prepared with the help of language experts. In this paper, we have developed POS taggers for Sindhi Language (in Devanagari Script), a resource poor language, using HMM (Hidden Markov Model) and Conditional Random Field (CRF).Evaluation results demonstrated the accuracies of 76.60714% and 88.79% in the HMM, and CRF, respectively.

1 Introduction

The main aim of NLP is to create suitable algorithms for computers to understand human language. These language technology tools help the users in translation and understanding of languages. For example, with the help of machine translation, a person can communicate or share information in their native language. Although plenty of different language processing tools are available for some languages, still some of the languages have not been able to attract the attention of the research community. The Sindhi language is one of them. There are 22 official languages of India. Sindhi is one of them. In 1967 it was recognized as an official language of India. The Eighth Schedules of the constitution of India includes the languages which are resource-poor and need to preserved and developed. Sindhi was included in this schedule in the 21st Amendment. Although in India, Sindhi is officially recognized, but it is not an official language of any of the states in India.

The "International Education System" puts a big problem for Sindhi people. The upcoming Sindhi generation is unable to write and speak the Sindhi language, as their parents do not communicate with them in the Sindhi language. This may lead to the extinction of the Sindhi language. For language preservation, our prime need is to save its speakers by developing some language processing tools which help them to learn Sindhi. In the last few years, some work has been done in the Sindhi language in Arabic script. Sindhi Devanagari is more resource-poor than Sindhi Arabic. In this research work we have developed a SLP (Sindhi Language Processing) tool i.e., POS Part of Speech Tagger in Devanagari Script.

The words can be classified in various lexical categories, such as nouns, verbs, etc. These categories are also known as Parts of Speech. Parts of Speech define their morphological and syntactical behavior. POS Tagging is a task of classifying each word in a corpus to a given syntactic class such as noun, verb, etc.

A word can belong to more than one lexical category, depending on its use in a sentence. The main objective of the POS tagging process is to remove this ambiguity. Tagger uses the contextual information to assign the tag. Part of Speech tagging is used in various applications of NLP, such as Machine Translation, Information retrieval, information extraction, spelling correction, and word sense disambiguation. This paper presents the development of two automatic taggers using Conditional (CRF) and Hidden Markov Model(HMM) .

2 Related work

POS tagging is assigning the syntactic or lexical category to a word in a sentence. POS tagging is a

Proceedings of the 18th International Conference on Natural Language Processing, pages 611–618
Silchar, India. December 16 - 19, 2021. ©2021 NLP Association of India (NLPAI)

fundamental task of NLP. It is an important pre-processing task of various Natural Language Processing (NLP) applications such as in IR, Text summarization, machine translation etc. In this section, we will discuss the related work done in this area.

2.2.1 Other Languages

The work on automatic POS tagging was started in the early 1960s. Ekbal, A. et al. (2007) developed a POS tagger for Bengali language using condition random field approach with a tag set of 26 POS tag. For training they used 72,341 words and 20 thousand words for testing. They got the accuracy of 90.3%.

Hasan, M. F. et al. (2007) applied few stochastic approaches such as unigram, bigram HMM and Brills POS tagging on Hindi, Bangla and Telugu with different size of the corpus. They found that Brill's transformation-based tagger's performance is good in comparison to other approaches.

Singh, T. D. et al. (2008) developed a POS tagger for Manipuri text using an unsupervised learning approach CRF. The system gave the Recall of 70% precision of 77.78% and F-measure of 73.68%.

Sharma et al. (2011) used an HMM algorithm to improve the accuracy of existing Punjabi POS Tagger. This Bi-gram tagger resolves the problem of ambiguity for complex and compound sentences. They have taken the training corpus of 20,000 tokens and a test corpus of 26 479 tokens. They achieved 90.11% accuracy.

Garrette, D et al. (2013) discussed the various aspects of semi-supervised Learning of POS taggers. They work for Kinyarwanda and Malagasy two resource-poor languages and study the effect of various kind of data on POS- tagger.

Singh, J., Joshi, N., & Mathur, I. (2013) used Statistical approach to develop Marathi POS tagger, i.e. Unigram, Bigram, Trigram and HMM. They achieved 77.38% accuracy for Unigram approach, 90.30% for Bigram, 91.46% for Trigram and 93.82% for HMM.

Sunitha, C. (2015) research work proposed a hybrid approach for POS tagging (CRF and rule-based approach) of Malayalam language. They used the tag set developed by IIIT Hyderabad. They got 94% accuracy.

Pakray, P. et al. (2015) developed various resources for Mizo language (an official language of Mizoram State) such as Mizo-to-English dictionary; tag set consist of 24 items and POS tagger.

Buys et al. (2016) proposed a model, which uses a Wsabie, a discriminative embedding model train a morphological tagger. They evaluated this on 11 languages and concluded that this model performs very well when used for closely related languages.

A CRF based approach was used by Sarkar, K. (2016) for developing POS Taggers for three language pairs, i.e. Bengali-English, Telugu-English and Hindi- English. They have got an average of 79.99 F1 scores.

For Odia Language, a CRF++ based POS tagger was developed by Behera, P. (2017). For this, they manually prepared POS annotated, the corpus of 600thousand tokens, using BIS tag set. The tagger is trained on 2,36,793 tokens and tested with 1,28,646 tokens. They got 94.39% accuracy for the known data and 88.87% accuracy for unknown data.

Mishra, P., Mujadia, V., & Sharma, D. M. (2018) presented an approach for POS tagging of resource poor language. This approach requires only the bilingual corpora of sentences. They have transferred the features of the resource-rich language to resource-poor language, for this they have used word alignment algorithm using Giza ++.

2.2.2 Sindhi language

Maher and Memon (2010 A) developed a POS Tagger using Word Net approach. This tagger was tested on lexicon containing 26,366 tagged words, and the accuracy was 97.14%. This Tagger gave higher accuracy on the past and presented tense sentences, but on future tense sentences, it gave lesser accuracy.

Maher and Memon (2010 B) developed the first POS tagging system for Sindhi (Perso-Arabic Script). They used a rule-based approach. The size of the lexicon was 26,366 and useda tag set of size 67. They tested this tagger on 1,500 sentences, which consisted of 6,783 words and obtained 96.28% accuracy.

Motlani et al. (2015) built a POS Tagger for Devanagari Script of Sindhi language using Conditional Random Fields. They tested and trained the tagger using 10-fold cross-validation. They used BIS tag set, and the accuracy of tagger was 92.6%.

3 The Approach

POS Tagging approaches are broadly classified into rule-based, stochastic, and hybrid approaches. In the rule-based approach, handwritten disambiguation linguistic rules are used for tagging. Stochastic is also known as a data driven approach, which requires pre tagged corpus for training. Hybrid is a combination of rule-based and data driven. To develop a POS tagger, we have chosen the stochastic approach. This is a data driven approach. Rule based approach is time consuming and needs language expertise to write the rule. For morphologically rich language, it is impossible to write all the rules

Stochastic approach is a probability-based approach. We have used two standard algorithms HMM (Hidden Markov Model) and CRF (Conditional Random Field) for POS tagging. HMM is an example of a Generative model, whereas CRF is an example of Discriminative model (Sutton, C., & McCallum, A. (2012)). For a particular data set, we cannot predict in advance which model will give the correct results. Each type of model is having its own limitations and delimitation.

4 Corpus Annotation

The stochastic approach is data driven. This requires the manually annotated corpus for training. The stochastic approach gives better results when the manually annotated corpus is used for training. In this sequence, we have manually annotated the corpus of 30000 sentences by using the guidelines described by Lata et al. (2012). For tagging, we have used the IL tag set.

A tag set is a collection of tags used by a tagger. The tag set is described in the following tables:

S. No.	**Tag**	**Description**	**Example**
1.	NN	Common Nouns	किताबु,माणहू
2.	NNP	Proper Nouns (Name of Person)	भारत, देहरादून
3.	NST	Noun Denotating Spatial and Temporal Expression	अगियां(आगे), पुठियां(पीछे)
4.	PRP	Proper Noun	अव्हांजे(आपकी), असांजे(हमारे)
5.	VM	Verb Main	थी(हो), रखण(रखना)
6.	VAUX	Verb Auxiliary	आहे(है), सघंदा(सकते)
7.	JJ	Adjective (Modifier of Noun)	कमज़ोर(कमजोर),तेज़(तेज)
8.	RB	Adverb	धीरे, जल्दी
9.	PSP	Post Position	खां(से),जे(के)
10	RPD	Particles	बि(भी),त (तो)
11	QTF	Quantifiers	घटि (कम), रुगो(केवल)
12	QTC	Cardinals	हिक(एक), ब(दो)
13	CCD	Conjunctions	पर(बल्कि), ऐं(और)

14	INTF	Intensifier	वधीक(अत्यधिक), तमाम(बहुत)
15	NEG	Negative	ननथा(नहीं)
16	SYM	Symbol	$, &, *, (,)
17	ECH	Echo Words	हलको - फुलको(हलका/JJ फुलका/ECH)
18	QO(QTO)	Ordinals	पहिरियों(पहला),बियनि (दूसरे) टिएं(तीसरे)
19	DMI	Demonstrative (Indefinite)	बियनि(किसी),कंहिं(किसी)
20	CCS	Subordinator	यदि(अगर),याने(अर्थात)
21	PRF	Pronoun (Reflexive)	ख़ुदि(खुद),पंहिंजी(अपनी)
22	DMD	Demonstrative (Deictic)	इहो, ही (यह),इनजो(इसका)

Table 1: Tag set for Sindhi Devanagari.

5 POS Tagging using HMM

For a given input sentence, we can calculate the best tag sequence using the following formula:

$$T' = \text{argmax}_T P(W/T)^* P(T) \quad (1)$$

Where P(T) is a prior probability of tag sequence (i.e., tag transition probability), and P(W/T) is emission probability. P(T) is calculated by using following formula:

$$P(T) = P(t_1)^* P(t_2/t_1)^* P(t_3/t_1t_2) \ldots *(t_n/t_1\ldots t_{n-1}) \quad (2)$$

According to bigram assumption:

$$P(t_i/t_i - 1) = \frac{c(t_{i-1}, t_i)}{c(t_{i-1})} \quad (3)$$

Where $c(t_{i-1},t_i)$ is the counting of how many times tag t_i, comes after tag t_{i-1} (Previous tag). To calculate the emission probability:

$$P(W/T) = P(w1/t1) * P(w2/t2)\ldots P(wi/ti) * \ldots P(Wn/tn) \quad (4)$$

$$\prod_{i=1}^{n} p(w_i t_i) \quad (5)$$

$$P(w_i / t_i) = \frac{c(w_i, t_i)}{c(t_i)} \quad (6)$$

Where $c(w_i/ t_i)$ calculate the probabilities , that a given tag t_i is associated with given word w_i.HMM algorithm chooses the most likely tag sequence with the help of a decoding algorithm, i.e. Viterbi algorithm.

5.1 Experiments

We have implemented the above algorithm in Python. Following table shows the Statistics of Training and Testing Data set:

Data Set	Number of Sentences
Training Data Set	30000
Testing Data Set	1000

Table 2: Data set for Experiment

5.2 Evaluation

Evaluation of output text is done by using the following formula:

$$\text{Accuracy} = \frac{\text{Total Number of correct POS tag generated by POS Tagger}}{\text{Total Number of POS Tag}}$$

We have tested this POS Tagger with 1000 sentences, which consists of 15680 tokens and we found 12012 matches. So, the overall accuracy obtained is 76.60714%.

6 POS Tagging using CRF

CRF was introduced by Lafferty et al., 2001.CRF is a discriminative undirected graphical model that belongs to the family of condition distribution. CRF is the most popular method used for structured prediction in the NLP task. In the discriminative approach, for a given input x and output y, the probability is calculated directly p(y|x) whereas in the generative model joint probability p(x,y) is generated. Structured means that the output of an algorithm is a structured object such as a tree or a sequence.
CRF is used for the POS tagging task. This discriminative model x for a given observation sequence O=<o1, o2, o3…. oT> where observation is the sequence of tokens, and State sequence S=<s1, s2, s3……sT> is the POS tag. Conditional probabilities are calculated as:

$$P\wedge(s/o)=\frac{1}{Z_0}\exp(\sum_{t=1}^{T}\sum_{k}\lambda k f k(st-1,st,0,t) \tag{7}$$

In the above equation, f_k is a transition feature function, which is learned via input or observation sequence. The weight of this function is k which defines the weight which is learned in training.

The normalization factor Z_0 is calculated using the following formula:

$$Z_o=\sum_{s}\exp(\sum_{t=1}\sum_{k}kfk(s_t-1,s_t,o,t)) \tag{8}$$

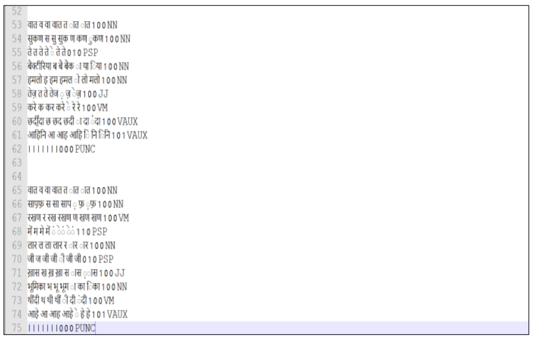
This makes all conditional probabilities sum equal to 0.

We have used CRF++ [1]for training and testing our tagger. Three files are required for CRF implementation.1 Training file 2. Testing File 3. Template File. The complete corpus is divided into 80-20 ratio, where 80% is used for training data, and 20 % is used for testing data. CRF model is developed in three steps:

6.1 Creation of Training and Testing File

All the words or token of a sentence must be represented using one token per line format.

[1] http://taku910.github.io/crfpp/



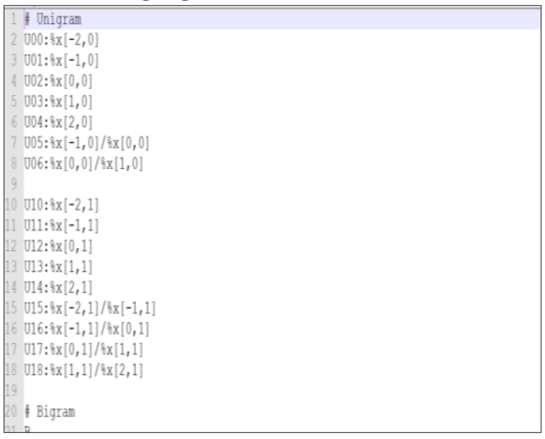
```

वात व वा वात त ा त ात 1 0 0 NN
सुकण स सु सुक ण कण ुकण 1 0 0 NN
ते त ते ते े ते ते 0 1 0 PSP
बेक्टीरिया ब बे बेक ा या िया 1 0 0 NN
हमलो ह हम हमल ो लो मलो 1 0 0 NN
तेज़ त ते तेज ़ ज़ ेज़ 1 0 0 JJ
करे क कर करे े रे रे 1 0 0 VM
छदींदा छ छद छदी ा दा ंदा 1 0 0 VAUX
आहिनि आ आह आहि ि नि िनि 1 0 1 VAUX
। । । । । । । 0 0 0 PUNC


वात व वा वात त ा त ात 1 0 0 NN
साप़फ़ स सा साप ़ फ़ ़फ़ 1 0 0 NN
रखण र रख रखण ण खण खण 1 0 0 VM
में म मे में ं ें ें 1 1 0 PSP
लार ल ला लार र ार ार 1 0 0 NN
जी ज जी जी ी जी जी 0 1 0 PSP
ख़ास ख ख़ ख़ा स ास ़ास 1 0 0 JJ
भूमिका भ भू भूम ा का िका 1 0 0 NN
थींदी थ थी थीं ी दी ंदी 1 0 0 VM
आहे आ आह आहे े हे हे 1 0 1 VAUX
। । । । । । । 0 0 0 PUNC
```

Figure 1: Sample Training File

Sentence boundary is identified by putting an extra blank line. Each token is represented along with its features in fixed columns, which are separated by space. First Column represents the word or token, and the last column represents the output on which we train CRF. The remaining columns represent the value of the features we have used in CRF. The sample training and the testing file are shown in the figure 1.

6.2 Creation of Feature Template

The template file defines the features used in CRF. Each line in a file represents one template. A macro used in a template will specify the token in input data, r specifies the row number from current token and c specifies the absolute position of the column. The sample of template is shown in the following figure.



```
# Unigram
U00:%x[-2,0]
U01:%x[-1,0]
U02:%x[0,0]
U03:%x[1,0]
U04:%x[2,0]
U05:%x[-1,0]/%x[0,0]
U06:%x[0,0]/%x[1,0]

U10:%x[-2,1]
U11:%x[-1,1]
U12:%x[0,1]
U13:%x[1,1]
U14:%x[2,1]
U15:%x[-2,1]/%x[-1,1]
U16:%x[-1,1]/%x[0,1]
U17:%x[0,1]/%x[1,1]
U18:%x[1,1]/%x[2,1]

# Bigram
```

Figure 2: Sample Template File

6.3 Training and Testing of CRF Model

For training the command is:

crf_learntemplate_filetrain_filemodel_file

Where the template file and train file which we have created in the previous step. For testing the command is:

% crf_test -m model_filetest_files ...

6.4 CRF Model Feature

We have created a CRF model which includes the following features:

- **Word length**: All the inflected words belong to open category that includes verbs and nouns. Inflection makes them lengthy. We included this feature as a binary feature. We have taken a word length of 3. If the word length exceeds 3 characters, we set the value 1 other wise 0.

- **Contextual information**: The task of the POS tagger is to assign a correct syntactical category to a word. Some words are ambiguous in the corpus. For example, the word “Book” can be used as a Noun or Verb. To resolve this ambiguity, the POS tagger will use the context, i.e. the preceding word and the following word. We have used the context window of size 5, i.e. current word and two previous and two following words.

- **Auxiliary verbs**: Auxiliary verb belongs to a closed category. We have prepared the list of 30 most frequently used auxiliary verbs. This feature is included as a binary feature if the token belongs to this list set the value of this feature 1 otherwise 0. Following are the few examples of Auxiliary verbs:

घुरिजे	चाहिए
वेंदियूं	जाती
आहे	है
वेंदा	जाते

- **Postposition**: Post positions are the most frequently used token in the sentence. They also belong to a closed category. We have identified the 11 most frequently used post position. This feature is also used as a binary feature. Following are the few examples of Post Position.

वटां(पाससे), वांगुर(तरह)

ते (पर), तां (ऊपरसे)

- **Affix, i.e. prefix and suffix:** Sindhi is a morphologically rich language, i.e. various forms of words are present. We can make various word forms, using affixation, i.e. by adding suffix and prefix. We have taken the length of the prefix 3 characters. It is proved that the length of 3 characters gives the best results (Motlani et al. (2015)). We have used a different combination of prefix and suffix length to train the tagger for morphology.

- **Postposition**: Post positions are the most frequently used token in the sentence. They also belong to a closed category. We have identified the 11 most frequently used post position. This feature is also used as a binary feature. Following are the few examples of Post Position.

वटां(पाससे), वांगुर(तरह)

ते (पर), तां (ऊपरसे)

6.5 Results

We have developed six CRF models. These models are evaluated using the aforementioned formulas. The following table shows the overall accuracy of various CRF model for each Tag. A final CRF model CRF_M7 is developed with all features and got 88.79% accuracy.

CRF Model	Features	Accuracy (%)
CRF_M0	No Feature	82.92
CRF_M1	Context	85.53
CRF_M2	Post Position	85.85

CRF_M3	Auxiliary Verb	85.96
CRF_M4	Length	86.34
CRF_M5	3 Prefix	96.93
CRF_M6	3 Suffix	97.05
CRF_M7	All Above Features	88.79

Table 3: Overall accuracy of CRF Model

7 Conclusion and Future Work

POS tagging is an important prerequisite task for any NLP research. In this research work we have developed two taggers for a resource poor language Sindhi in Devanagari script, using stochastic approach. For this we have used 1.HMM and 2.CRF. First we have manually annotated the corpus of 30000 sentences. We have got accuracy of 76.60714% and 88.79% for HMM and CRF respectively.

Sindhi is a morphologically, rich language. The various morphological features such as prefix, postfix and word length will help in defining the POS of a word. CRF can incorporate all these features in the model, and this is one of the main advantages of CRF over HMM. In future to handle the exceptional cases, we could merge the rule-based tagger with the statistical approach. We could use various other machine learning approaches such as SVM to develop the POS tagger. In addition we can develop other tools such as Named Entity Recognizer (NER) which will increase the accuracy of tagger. A robust tagger could be developed using an ensemble approach.

Acknowledgments

We would like to acknowledge lot of people for their help and support. Dr. Haso Dadlani (Ret. Head Department of Sindhi Govt. college Ajmer) for helping us in creating and annotating the data. Dr. Murlidhar Jetley for providing all the necessary Sindhi Grammar books.

References

Behera, P. (2017). An Experiment with the CRF++ Parts of Speech (POS) Tagger for Odia. Language in India, 17(1).

Ekbal, A., Haque, R., & Bandyopadhyay, S. (2007, December). Bengali part of speech tagging using conditional random field. In Proceedings of Seventh International Symposium on Natural Language Processing (SNLP2007) (pp. 131-136).

Garrette, D., Mielens, J., & Baldridge, J. (2013). Real-world semi-supervised learning of POS-taggers for low-resource languages. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (Vol. 1, pp. 583-592).

Hasan, M. F., UzZaman, N., & Khan, M. (2007). Comparison of Unigram, Bigram, HMM and Brill's POS tagging approaches for some South Asian languages.

I. I. Ayogu, A. O. Adetunmbi, B. A. Ojokoh and S. A. Oluwadare, "A comparative study of hidden Markov model and conditional random fields on a Yorùba part-of-speech tagging task," 2017 International Conference on Computing Networking and Informatics (ICCNI), 2017, pp. 1-6, doi: 10.1109/ICCNI.2017.8123784.

Lata, S., Chandra, S., Verma, P. and Arora, S. (2012). Standardization of POS Tag Set for Indian Languages Based on XML Internationalization Best Practices Guidelines. Proceedings of LREC (WILDRE) First Workshop on Indian Language Data: Resources and Evaluation. Istanbul, Turkey. 01-17.

Mahar, J. A., &Memon, G. Q. (2010 A). Sindhi Part of Speech Tagging System using WordNet. International Journal of Computer Theory and Engineering, 2(4), 538.

Mahar, J. A., &Memon, G. Q. (2010, B). Rule Based Part of Speech Tagging of Sindhi Language. In Signal Acquisition and Processing, 2010. ICSAP'10. International Conference on (pp. 101-106). IEEE.

Mishra, P., Mujadia, V., & Sharma, D. M. (2018). POS Tagging For Resource Poor Indian Languages Through Feature Projection.

Motlani, R., Lalwani, H., Shrivastava, M., & Sharma, D. M. (2015). Developing Part-of-Speech Tagger for a Resource-Poor Language: Sindhi. In Proceedings of the 7th Language and Technology Conference (LTC 2015), Poznan, Poland.

Pakray, P., Pal, A., Majumder, G., & Gelbukh, A. (2015, October). Resource building and parts-of-speech (POS) tagging for the Mizo language. In 2015 Fourteenth Mexican International Conference on Artificial Intelligence (MICAI) (pp. 3-7). IEEE.

Sutton, C., & McCallum, A. (2012). An introduction to conditional random fields. Foundations and Trends® in Machine Learning, 4(4), 267-373.

Sharma, S. K., &Lehal, G. S. (2011, June). Using hidden markov model to improve the accuracy of Punjabi pos tagger. In 2011 IEEE International Conference on Computer Science and Automation Engineering (Vol. 2, pp. 697-701). IEEE.

Singh, J., Joshi, N., & Mathur, I. (2013, August). Development of Marathi part of speech tagger using statistical approach. In 2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (pp. 1554-1559). IEEE.

Singh, T. D., Ekbal, A., & Bandyopadhyay, S. (2008). Manipuri POS tagging using CRF and SVM: A language independent approach. In proceeding of 6th International conference on Natural Language Processing (ICON-2008) (pp. 240-245).

Sunitha, C. (2015, August). A hybrid Parts of Speech tagger for Malayalam language. In 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (pp. 1502-1507). IEEE

Stress Rules from Surface Forms: Experiments with Program Synthesis

Saujas Vaduguru[1] **Partho Sarthi**[2] **Monojit Choudhury**[3] **Dipti Misra Sharma**[1]
[1] IIIT Hyderabad [2] University of Wisconsin, Madison[*]
[3] Microsoft Research India
[1] {saujas.vaduguru@research.,dipti@}iiit.ac.in
[2] sarthi@wisc.edu
[3] monojitc@microsoft.com

Abstract

Learning linguistic generalizations from only a few examples is a challenging task. Recent work has shown that *program synthesis* – a method to learn rules from data in the form of programs in a domain-specific language – can be used to learn phonological rules in highly data-constrained settings. In this paper, we use the problem of phonological stress placement as a case to study how the design of the domain-specific language influences the generalization ability when using the same learning algorithm. We find that encoding the distinction between consonants and vowels results in much better performance, and providing syllable-level information further improves generalization. Program synthesis, thus, provides a way to investigate how access to explicit linguistic information influences what can be learnt from a small number of examples.

1 Introduction

Deep neural models have driven recent success in NLP, including models for tasks in phonology and morphology. They have been applied to tasks such as grapheme-to-phoneme conversion (Ashby et al., 2021) and morphological reinflection (Pimentel et al., 2021), and also been incorporated into theories of phonology (Wu et al., 2021). While neural models are powerful learning machines, they require a large number of training examples, either for supervised or for transfer learning. Additionally, these models are not easily interpretable, and understanding what stuctures and patterns these models learn from data is a non-trivial task.

In this paper, we explore an different approach to learning linguistic patterns from data – *program synthesis*. Program synthesis (Gulwani et al., 2017) is a method to learn interpretable rules in the form of programs in a *domain-specific language* (DSL). The design of the DSL allows for specifying domain information, such as various linguistic concepts, and using these to guide learning from data.

Word	Stress pattern
sata	0100
hiha	0100
vatova	000100
kahasi	000100
ʔaona	01000
dehiaʔhe	00001000

Table 1: An example of the task of predicting stress patterns based on surface forms from the Cofan language. Each phoneme in each word is labelled with 1 for primary stress or 0 for secondary stress.

Vaduguru et al. (2021) show that program synthesis can be used to learn linguistic rules from a small number of examples, and apply it to learning phonological rules that perform string-to-string transformations. They demonstrate their method for learning different types of rules, including morphophonology, transliteration, and phonological stress.

In this paper, we investigate how the design of the DSL influences what rules are learnt from data. To do this, we focus on learning rules that determine placement of phonological stress from data. Phonological stress depends on both the position of a syllable within words, and language-dependent syllable weight hierarchies. This allows us to study how encoding information about position within a word and distinctions relevant to syllable weight hierarchies affects a program synthesis system designed to learn these rules from only a small number of examples.

We extend the formulation of phonological stress placement as a string-to-string transformation prob-

[*]Work done while at the Microsoft Research India

lem from Vaduguru et al. (2021) and develop a program synthesis approach specific to stress. We design different DSLs, each providing access to different phonological abstractions. We compare the results from using these different DSLs on data from a variety of languages.

Through the example of using program synthesis to learn stress rules, we seek to illustrate how program synthesis can be used as a general framework to compare how providing the same learning algorithm access to different linguistic abstractions can influence generalization from some given data.

2 Program Synthesis

Program synthesis is the task of finding a program in a domain-specific language (DSL) that satisfies certain constraints (Gulwani et al., 2017). This approach allows us to encode domain-specific assumptions about a task, and use generic search-based (Polozov and Gulwani, 2015) or constraint-based (Solar-Lezama, 2008) approaches to synthesize programs.

We apply program synthesis to learn rules for stress placement. The programs that are synthesized operate directly on the surface form of a word, which is provided as a string. By varying the structure of the DSL, we control the kind of phonological abstractions that are available to the synthesizer, and observe the effects of this on learning and generalization.

2.1 Stress rules as programs

We model stress rules as string-to-string transformations. Formally, we synthesize program that implements a function $f : \Sigma^* \rightarrow \{0, 1, 2, 3\}^*$, where Σ is the set of phonemes in a language. f takes as input a sequence of phonemes $w_1 w_2 \ldots w_n$, and assigns a "degree of stress" to each phoneme. 0 indicates that a phoneme is unstressed, 1 indicates primary stress, 2 secondary stress, and 3 tertiary stress. Since stress is applied at the level of the syllable, we conventionally mark the first vowel of a syllable with the degree of stress, treating it as the 'locus' of stress within a syllable. We refer to this output string composed of the degree of stress for each phoneme in the input word as the *stress pattern* for the word.

The programs we synthesize take the form of sequences of rules similar to rewrite rules (Chomsky and Halle, 1968).

Each rule is of the form

$$\phi_{-l} \cdots \phi_{-1} X_1 \cdots X_c \phi_1 \cdots \phi_r \rightarrow T \qquad (1)$$

A rule applies to a central phoneme which satisfies a conjunction of predicates $X_1, \ldots, X_c$, which appears in a context defined by the conjunction of predicates $\phi_{-l}, \ldots, \phi_{-1}$ (which apply to l phonemes to the left of the central phoneme) and $\phi_1, \ldots, \phi_r$ (which apply to r phonemes to the right of the central phoneme). If the conjunction of all predicates is satisfied, a transformation T is applied to the phoneme.

The DSL defines the set of predicates $\mathcal{P}$ and set of transformations $\mathcal{T}$ that can be used. We vary the predicates ($\mathcal{P}$) available to the synthesizer to define different DSLs, each providing access to a different classes of phonological abstractions. The transformation is a function that takes the phoneme as input and outputs the degree of stress, and is of the form `ReplaceBy(`s`)`, where s is a value representing the degree of stress.

2.2 Domain-specific languages

A domain-specific language is a declarative language that defines the set of programs within which we need to search. It is defined by a set of operators, their semantics, and a grammar that defines rules to combine operators. Each operator also has an associated score, which can be combined with the scores for other operators in the program to derive a *ranking score* that can be used to break ties among multiple correct programs. By appropriately choosing the operators, their semantics, and the scores associated with them, we can control domain-specific knowledge and preferences available to the synthesizer.

We use a DSL that implements rules of the form described in Section 2.1 using if-then-else constructs. This allows us to define a sequence of rules, the application of which is conditioned on a conjunction of predicates. The first rule for which the condition is satisfied is executed. The sequence of rules is applied to each phoneme of the input to obtain the degree of stress on that phoneme. This is achieved using a `Map` operator.

As described in Section 2.1, the condition for the `IfThenElse` constructs is defined as a conjunction of predicates. Based on the set of predicates available to the DSL, we define a sequence of 4 DSLs, each of which provides access to a different set of phonological classes (sets of phonemes).

```
output := Map(rules, input_phonemes)
rules := IfThenElse(C, T, rules) | T
```

Figure 1: `IfThenElse` statements in the DSL. A transformation `T` is applied if the condition `C` is true, else a transformation determined by the remaining `rules` is applied.

A predicate is defined by a predicate type and a class of phonemes to which it applies. We define various classes, and use groups of these classes to define different DSLs.

2.2.1 Classes of phonemes

The most basic set of classes is the set of singleton classes, each referring to one phoneme. We then define classes of consonants and vowels. Most stress systems do not distinguish between different (short) vowels to determine syllable weight hierarchies. Allowing this distinction to be made can allow the synthesizer to learn rules that identify syllable types as a sequence of vowels and consonants in a specific order.

Phonemes that share phonological features are also grouped into classes. We include vowel features such as height and frontness, and also features of consonants such as place and manner of articulation.

Finally, we define classes based on syllable-level information, such as whether a phoneme is the first vowel of a long vowel, diphthong, open syllable, or closed syllable. For our synthesizer, we define these uniformly across languages. A diphthong refers to a sequence of two different vowels. We treat a syllable as closed when the vowel is followed by multiple consonants, and break the syllable after the first consonant. A syllable that is not closed is treated as open.

The classes that are available to each of the 4 DSLs we define – BASIC, CV, SYLLABLE, and FEATURE – are shown in Figure 2.

2.2.2 Predicate types

We define a number of predicate types, which determine the positions in the word to which the predicate applies. In each of these cases, `X` refers to a class of phonemes. We will illustrate how each of this predicate types, defined for one unit, can be used as part of a hypothetical stress rule.

`IsX` predicates determine whether a phoneme is a member of a particular class. For example, since consonants are not stressed, `IsConsonant` can be used to ensure the output at a consonant is 0.

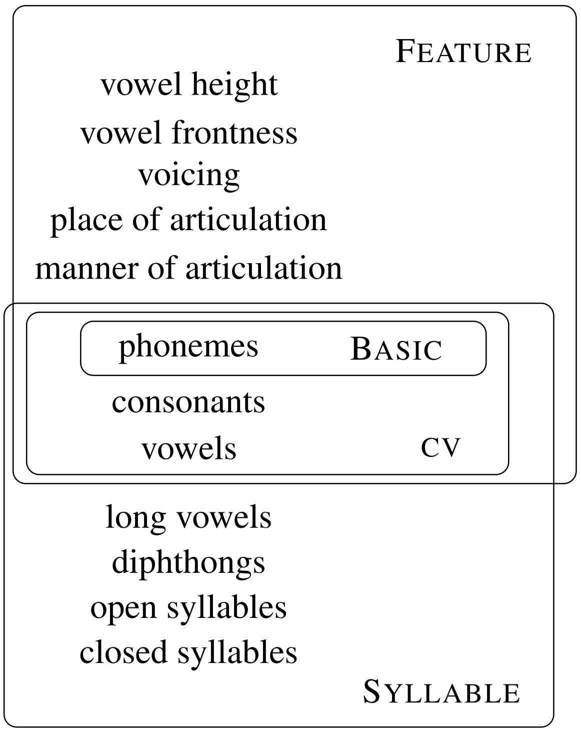


Figure 2: Classes available to each DSL – BASIC, CV, SYLLABLE, and FEATURE.

`IsKthX` predicates take an additional argument K, and determine if a phoneme is the K^{th} occurence of a member of a class in the word. If a stress rule places primary stress on the second closed syllable of a word, then the `IsKthClosedSyllable` predicate can be used with $K = 2$ to select that syllable. Note that K can also be negative, to refer to units counting from the right edge of the word.

Each of these predicates may apply to either the central phoneme (one of the X_i from eq. (1)), or phonemes in the context (one of the ϕ_i from eq. (1)). We guide the synthesizer to prefer simpler rules by ranking predicates that refer to nearby phonemes (at a smaller displacement from the central phoneme) above those that refer to more distant phonemes. For `IsKthX` predicates, we rank predicates that take a smaller absolute value of K higher to guide the synthesizer to prefer rules that refer to edges of the word over arbitrary positions in between. We also define additional types of predicates that can refer only to the central phoneme.[1]

`PrefixContainsX` predicates check whether the prefix of the word up to, but not includ-

[1]These predicates are not included in the FEATURE DSL due to the requirement of enumerating a very large number of predicates.

ing, the phoneme contains any instances of a class. If a stress rule places primary stress on the first occurrence of /e/ in a word, then `PrefixContainsPhoneme(e)` can be used to ensure other occurrences of /e/ are not stressed.

`SuffixContainsX` predicates check whether the suffix of the word after, but not including, the phoneme contains any members of a class. Similar to the example above, if the last occurrence of /e/ is to be stressed, `PrefixContainsPhoneme(e)` can be used to ensure other occurrences are not stressed.

`WordContainsX` predicates check whether a member of the class exists anywhere in the word. If a stress rule places stress on the first vowel of the word if there are no long vowels in the word, then `WordContainsLongVowel` can be used to ensure stress is not placed on the first vowel incorrectly.

2.3 Synthesis algorithm

Synthesis begins with extracting phoneme-aligned pairs from the words. Each example is a pair of a phoneme and the degree of stress with which it is labelled. The synthesis algorithm then learns rules that map a phoneme (in its context) to the correct label.

To synthesize `IfThenElse` constructs, we adapt the LearnProgram, LearnBranch, and LearnConj procedures from Kini and Gulwani (2015). These procedures allow for learning *decision lists*, which are sequences of predicate-transformation pairs of the form $\langle (p_1, t_1), (p_2, t_2), \ldots, (p_n, t_n) \rangle$, where each p_i is a conjunction of atomic predicates introduced before, and t_i is a transformation function. The list is constructed such that given a set of examples X, the set can be partitioned into n subsets such that for the i^{th} subset X_i is does not satisfy any of the predicates $p_1, \ldots, p_{i-1}$, and satisfies p_i, and the transformation t_i results in the correct output for the examples in X_i. These decision lists correspond to nested `IfThenElse` constructs. An example is tested for the predicate p_i. If the predicate is true of the example, t_i is executed, and execution is terminated. If not, the else clause – which represents the rest of the list – is executed.

The LearnProgram procedure learns a decision list given a set of input-output examples X, optimizing for a shorter list. The procedure maintains a set R of examples which haven't yet been covered by any of the predicates of the decision list, which is initialized with the entire set X. The procedure then calls LearnBranch, which learns the next element of decision list – a predicate that determines when the item will apply, and a corresponding action. Examples which satisfy the predicate are then removed the from R. This is repeated till R is empty.

LearnBranch starts by generating a set of candidate transformations. Each transformation divides the set of examples into two – those it transforms correctly, and those it does not. Then, the LearnConj can be used to obtain conjunctions of candidate atomic predicates that are true for the most examples in the former set, and false for all examples in the latter set. The conjunctions with the best ranking scores (determined as the sum of the scores for individual atomic predicates) are then each combined with the transformation to obtain predicate-transformation pairs. The predicate-transformation pair that covers the largest number of examples is then chosen as the next element of the decision list.

The LearnBranch and LearnConj procedures require the synthesis of candidate atomic predicates and transformations. These are synthesized using the FlashMeta algorithm. Given the transformation or predicate operator (as described in Section 2.2), FlashMeta can be used to infer arguments to the operator such that it satisfies a given set of examples. Based on the examples, FlashMeta finds the position of phonemes to which a predicate applies relative to the central phoneme (a value between $-l$ and r in eq. (1), where 0 refers to the central phoneme), and values of additional arguments to the predicate such as the value of K in predicates of the type `IsKthX`. FlashMeta also finds the output value for transformation operators. To do this, FlashMeta uses the *inverse semantics* of the operators, which constrains the values of arguments given the behaviour of the operator as input-output examples. Figure 3 illustrates the working of the synthesis algorithm.

3 Dataset

We obtain data by consulting grammars and other linguistic and phonological analyses of languages listed in the StressTyp2 database (Goedemans et al., 2014) or by Gordon (2002). The database contains information about various lects and the kinds of stress patterns exhibited by these lects. The database also has links to the sources from which the data was collected for compiling the database, and these were the sources we consulted

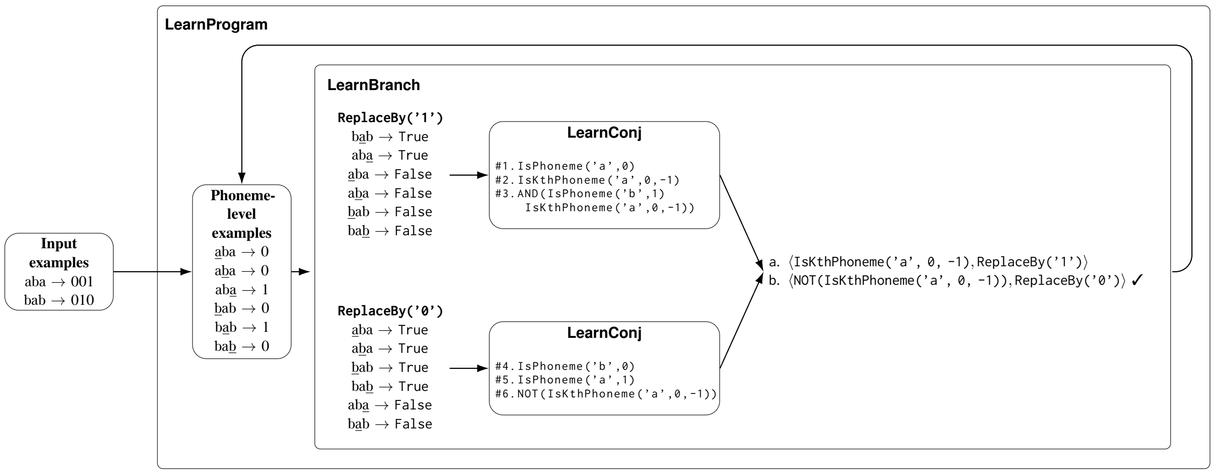


Figure 3: Illustration of the synthesis algorithm on a hypothetical case where the stress is on the last vowel, using the BASIC DSL. The input examples are first used to generate phoneme-level examples. The LearnProgram procedure then learns a decision list for the phoneme-level examples through calls to LearnBranch. The LearnBranch procedure iterates through different candidate transformations (such as ReplaceBy('0') and ReplaceBy('1')). For each transformation, the LearnConj procedure produces candidate conjunctions for when the transformation applies and when it does not. The candidate which is true for the most number of cases where the transformation applies, and none of the cases where it does not, is chosen. Here, this is #1 for the ReplaceBy('1') action and #6 for the ReplaceBy('0') action. The predicate-action pair which solves the most examples (here b) is then added to the decision list, and the LearnBranch procedure is called again on the unsolved examples.

for examples of words with stress patterns marked. All the words collected from these sources have the stress marking attested in the source – there are no cases of a given rule being used to predict the stress pattern on words.

Once words and the corresponding stress patterns are collected for a language, the set of words is split into two parts – one to be used for synthesizing programs (the *training* split) and the other (the *test* split) to be used for evaluating the synthesized rules. We ensure that all test examples are marked with a stress rule that is attested in the training examples.

We also use data from the stress problems presented in Vaduguru et al. (2021). These problems are chosen from the Linguistics Olympiads, a set of contests in linguistics for high school students. They present a task that is of a similar form to the tasks we study in this paper. These problems present words from a language with stress marked, and require a solver to use this data to infer the stress rules for that language, just as we do for the program synthesis system in this work. For these problems, we preserve the train-test splits from Vaduguru et al. (2021).

In total, we have data from 34 languages – 28 from the data we collect, and 6 from Linguistics Olympiad problems. Each language has between 5 and 33 training examples, with an average of 11.3, and between 2 and 16 test examples, with an average of 4.8.

4 Experiments

As described in Section 3, each language has a number of pairs – of word and stress pattern – in the training split. These are provided to the synthesis system, which produces a program. Given that the system checks shorter programs before longer ones, programs that are found after a long search are likely to be overfit to the given examples, and unlikely to generalize to unseen cases. This is why we terminate the synthesis if a program isn't found within 60 minutes. We also observe that for most of the languages, synthesis terminates well before this limit. For each language, we experiment with each of the 4 DSLs described in Section 2.2.

We also experiment with two neural sequence-to-sequence baselines, based on the LSTM and the Transformer architecture respectively. We use the implementation made available by Wu (2020), and train models with the same hyperparameters as in Vaduguru et al. (2021).

To evaluate the synthesized programs, we consider the output of the program on words in the test split. We only consider cases where the predicted stress pattern exactly matches the ground truth stress pattern as correct, and compute the fraction of samples for which the predictions are correct – the *accuracy* of the program on the test set. We report the average accuracy for the set of languages we consider. We also report the average accuracy separately for data from each source – data which we collect and that chosen from Linguistics Olympiads – to observe any differences based on the source of data.

Additionally, we report the number of languages for which a synthesizer acheives a test accuracy of 100% or over 50%. This allows us to count the number of languages for which the synthesizers infer all, or a substantial fraction of, the rules of stress placement.

4.1 Results

The results obtained are shown in Tables 2 and 3. Language-wise results are presented in Appendix A. As expected, we see that neural baselines achieve low scores (except LSTM models on data from the Olympiads). Using program synthesis allows for significant gains over these baselines.

We observe that providing no information beyond the identity of the phonemes is not sufficient to infer correct rules. This is seen in the low overall accuracy obtained using the BASIC DSL, and the fact that it doesn't achieve perfect test accuracy for any of the languages.

Providing the DSL with just the distinction between consonants and vowels results in a big jump in performance. The CV DSL achieves a much higher average test accuracy, and is able to infer the rules fully in a number of languages.

Since stress placement is determined based on syllables, it is not surprising that encoding distinctions relevant to syllable weight hierarchies, such as vowel length and open/closed-ness of syllables, achieves the best performance. The SYLLABLE DSL achieves the highest average test accuracy, and the infers the rules fully in the highest number of languages.

While providing access to other features of phonemes in the FEATURE DSL does improve upon providing only the consonant-vowel distinctions, we see that it does not help as much as providing access to syllable-level distinctions.

We also note the difference between different sources here. Since Linguistics Olympiad problems are intended as reasoning challenges where solvers have to infer rules, they pose a more difficult learning challenge for our program synthesis system. This is seen in the lower accuracy obtained using all the DSLs for these languages. We also note that in these problems, access to syllable-level distinctions provides a larger gains relative to access to only consonant-vowel distinctions.

5 Analysis

We examine the synthesized programs for specific languages to understand the reasons for different levels of performance when using different DSLs, and illustrate patterns in failures due to properties of the DSL.

5.1 Benefits of the consonant-vowel distinction

We see that providing the synthesizer access to the distinction between vowels and consonants can improve its performance significantly. A synthesizer that does not have access to these needs to infer from the data alone that different vowels may behave in the same way, and that the behaviour may be common in a variety of contexts. In the absence of a large amount of data to provide negative evidence that occurrence in a specific context determines the application of a rule, the synthesizer tends to discover incorrect rules.

Consider the example of Lezgian. Stress is Lezgian is always placed on the second syllable of a word. Using the BASIC DSL, the system discovers rules such as

```
IfThenElse(
 And(PrefixContainsPhoneme('a', v, i),
 And(PrefixContainsPhoneme('l', v, i),
     Not(
     IsKthPhoneme('f', 0, 0, v, i)
     ))),
    ReplaceBy('1'))
```

This rule places stress on a phoneme if the prefix of the word up to the phoneme contain /a/ and /l/, and it is not the first occurrence of /f/ in the word. It also learns the rule

```
IfThenElse(
    SuffixContainsPhoneme('i', v, i),
    ReplaceBy('0'))
```

which does not place stress on a phoneme if the phoneme /i/ occurs after it in the word. This would lead to incorrect predictions if /i/ occurs in the third syllable of the word. Such rules are clearly overfit to the training data, and do not generalize well.

Languages	BASIC	CV	SYLLABLE	FEATURE	LSTM	Transformer
All	18.9	46.4	60.8	52.8	15.0	12.7
– Ours	18.8	52.8	63.9	57.1	13.2	12.8
– Olympiad	19.4	16.7	46.1	32.2	23.2	12.1

Table 2: Average accuracy across languages for each of the different DSLs for the entire set of languages and grouped by source of data.

Languages	BASIC		CV		SYLLABLE		FEATURE	
	$= 100\%$	$\geq 50\%$	$= 100\%$	$\geq 50\%$	$= 100\%$	$\geq 50\%$	$= 100\%$	$\geq 50\%$
All	0	7	8	17	12	21	11	18
– Ours	0	6	7	16	11	18	10	17
– Olympiad	0	1	1	1	1	3	1	1

Table 3: Number of languages where the system obtains perfect test accuracy, or test accuracy over 50%.

On the other hand, with the CV DSL, just the rules

```
IfThenElse(
    IsKthVowel(0, 1, v, i),
    ReplaceBy('1'),
    ReplaceBy('0'))
```

are learnt, which place stress on a phoneme if it is the second vowel (indexing starts at 0), and does not in all other cases. This illustrates the importance of access to such phonological distinctions when rules need to be learnt from a small amount of data.

5.2 Benefits from syllable-level distinctions

The benefits of being able to refer to syllable-level information in rules is visible in the programs synthesized for Sio. Stress in Sio depends on the weight of the syllable. If the final syllable of the word is a heavy syllable, it is stressed. If not heavy, the penultimate syllable is stressed. One of the rules the SYLLABLE grammar learns is

```
IfThenElse(
    Not(SuffixContainsDiphthong(v, i)),
    ReplaceBy('1'))
```

While there are other constraints to placement, this rule works towards ensuring that if the final syllable contains a diphthong (which is part of a heavy syllable), it is not stressed incorrectly.

To infer a rule about diphthongs correctly within the CV DSL, predicates about the first vowel have to be taken in conjunction with predicates about the second vowel, and this conjunction has to be distinguished from many other competing conjunctions which may also be consistent with the data. If other conjunctions which don't generalize beyond the training data are simpler, these are ranked higher and incorrectly chosen. Allowing the DSL to distinguish concepts such as diphthongs thus allows for learning simpler rules in such situations.

5.3 Incorrect generalizations

However, providing access to syllable-level distinctions may also encourage the synthesizer to discover incorrect generalizations. We see this in the case of Tzutujil. Stress in Tzutujil is placed on the final syllable of a word. With the CV DSL, the following rules are learnt.

```
IfThenElse(
  And(SuffixContainsVowel(v, i),
  And(IsKthVowel(0, -2, v, i),
      IsKthVowel(1, -1, v, i))),
    ReplaceBy('1'))

IfThenElse(
  And(Not(SuffixContainsVowel(v, i)),
      IsKthConsonant(-1, 0, v, i)),
    ReplaceBy('1'))

IfThenElse(
  And(Not(SuffixContainsVowel(v, i)),
      IsKthVowel(0, 1, v, i)),
    ReplaceBy('1'))
```

The first rule checks that if the suffix of the word after a phoneme to be stressed contains a vowel, it is the last vowel in the word. This is a case where the rule for a diphthong is discovered in the CV DSL. The other two rules ensure that a non-final vowel is not stressed by checking that the suffix doesn't contain any vowels.

The SYLLABLE DSL on the other hand discovers the rule

```
IfThenElse(
    Not(IsOpenSyllableVowel(0, v, i)),
    ReplaceBy('1'))
```

which incorrectly places stress on any vowel that is part of an open syllable. This results in the SYLLABLE DSL performing worse than the CV DSL for Tzutujil.

5.4 Insufficient constraints for stress placement

A common reason for failure is the failure to learn sufficient constraints for the application of rules. This results in sets of rules which allow primary stress to be placed on multiple phonemes, or on no phonemes, both of which are incorrect. We see examples of this in the program synthesized for stress in Cofan, where the penultimate syllable of the word is stressed.

Using the CV DSL, the following are some of the rules that are synthesized.

```
IfThenElse(
 And(IsKthVowel(0, 1, v, i),
     PrefixContainsPhoneme('k', v, i)),
 ReplaceBy('1'))

IfThenElse(
 And(PrefixContainsPhoneme('s', v, i),
     IsKthVowel(0, 0, v, i)),
 ReplaceBy('1')
```

Neither of these rules are sufficiently general, and rely on the presence of /k/ or /s/ in the prefix of the word up to the phoneme, neither of which is not relevant to the placement of stress. However, another problem is that there is no constraint that prevent both these rules applying to the same word. This occurs for the Cofan word /sokɨ/. The program incorrectly predicts that both syllables in this word receive primary stress, which is not allowed.

The program synthesized with the SYLLABLE DSL for the same data includes the rule

```
IfThenElse(
    IsKthConsonant(-1, -2, v, i),
 ReplaceBy('1'))
```

This rule places stress on the phoneme following the penultimate consonant of the word. When the word ends with two open syllables, this rule correctly predicts stress. However, for a word such as /ʔaiʔpa/, this rule does not apply. When other rules also fail to apply, as is the case for this word, no phoneme is predicted to be stressed. This violates the requirement that at least one syllable receive primary stress.

6 Related work

6.1 Program synthesis for linguistics

Barke et al. (2019) and Ellis et al. (2015) present program synthesis as a method for learning morphophonological rules from examples. They assume the existence of underlying forms, and infer rules of inflection that map an underlying form to the surface form using program synthesis. These rules operate directly on features of the phoneme, and not on the surface form of the words.

Sarthi et al. (2021) apply program synthesis to the problem of grapheme-to-phoneme conversion in Hindi and Tamil, which they pose as a string-to-string transformation task. Their design of the domain-specific languages captures specific phonological processes in Hindi and Tamil.

Vaduguru et al. (2021) apply program synthesis to learning phonological rules for string-to-string transformations from a small number of examples. They show that the method can be used to learn rules for various phonological phenomena like morphophonological rules, phonological rules relating similar languages, and stress rules in a challenging set of problems drawn from the Linguistics Olympiads. In this work, we focus on learning rules for stress placement alone, which allows us to specialize the DSL and investigate the effect of encoding phonological knowledge in the DSL.

6.2 Learning rules of phonological stress

Dresher and Kaye (1990) develop a system that learns stress patterns within the principles and parameters framework. Given words and the structure of syllables in these words, their method learns the parameters for principles relevant to the placement of stress.

Gupta and Touretzky (1992) propose a perceptron-based method for learning stress rules for empirical data. They propose a model that takes as input the weight of a syllable and predicts a value corresponding to the type of stress on the syllable.

Heinz (2006) proposes a method to learn rules for quality-insensistive stress, where the stress pattern depends only on the position and not on the weight of a syllable. Using the property of neighbourhood-distinctness, they propose a method that learns a finite-state machine to model stress patterns.

While these works consider the learnability of stress patterns using a model that assumes certain features or properties of the input to be available, we propose a generic method where the availability of features can be controlled, and learning of abstract composite concepts like syllable weights from various primitive concepts can be investigated.

7 Conclusion

In this paper, we explore the problem of learning rules for the placement of phonological stress from only a few examples using program synthesis. We pose the problem as one of learning rules in the form of programs for string-to-string transformations. By designing the domain-specific language in which the rules are synthesized, we can control the amount of linguistic information available to the synthesizer.

We use the allowance to explicity provide the learning algorithm access to linguistic information to investigate how different linguistic concepts influence the rules that are learnt from data. To do this, we develop a generic program synthesis algorithm, and different domain-specific languages in which programs are synthesized. Each algorithm provides access to a different set of phonological classes, which can be used to identify phonemes that share common features.

We find that given a small number of examples, a synthesizer that doesn't have access to linguistic information beyond phoneme identity is unable to learn any useful rules. However, distinguishing consonants and vowels proves extremely useful, and distinguishing different types of syllables proves even more so.

Thus, using synthesis of rules for stress as a case study, we show how program synthesis can be used as a way to compare how different primitive concepts can be combined to learn rules for the same data using the same learning algorithm. Such methods can therefore be used to analyze what concepts are necessary to learn various rules from a limited number of samples, without changing the way in which these concepts are combined. Since program synthesis results in human-readable programs, we can also understand how primitive concepts are combined based on the data.

References

Lucas F.E. Ashby, Travis M. Bartley, Simon Clematide, Luca Del Signore, Cameron Gibson, Kyle Gorman, Yeonju Lee-Sikka, Peter Makarov, Aidan Malanoski, Sean Miller, Omar Ortiz, Reuben Raff, Arundhati Sengupta, Bora Seo, Yulia Spektor, and Winnie Yan. 2021. Results of the second SIGMORPHON shared task on multilingual grapheme-to-phoneme conversion. In *Proceedings of the 18th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 115–125, Online. Association for Computational Linguistics.

Shraddha Barke, Rose Kunkel, Nadia Polikarpova, Eric Meinhardt, Eric Bakovic, and Leon Bergen. 2019. Constraint-based learning of phonological processes. 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 6176–6186, Hong Kong, China. Association for Computational Linguistics.

Noam Chomsky and Morris Halle. 1968. The sound pattern of english.

B.Elan Dresher and Jonathan D. Kaye. 1990. A computational learning model for metrical phonology. *Cognition*, 34(2):137–195.

Kevin Ellis, Armando Solar-Lezama, and Josh Tenenbaum. 2015. Unsupervised learning by program synthesis. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Rob Goedemans, Jeffrey Heinz, and Harry Van der Hulst. 2014. Stresstyp2. *University of Connecticut, University of Delaware, Leiden University, and the US National Science Foundation*.

Matthew Gordon. 2002. A factorial typology of quantity-insensitive stress. *Natural Language & Linguistic Theory*, 20(3):491–552.

Sumit Gulwani, Oleksandr Polozov, and Rishabh Singh. 2017. Program synthesis. *Foundations and Trends® in Programming Languages*, 4(1-2):1–119.

Prahlad Gupta and David Touretzky. 1992. A connectionist learning approach to analyzing linguistic stress. In *Advances in Neural Information Processing Systems*, volume 4. Morgan-Kaufmann.

Jeffrey Heinz. 2006. Learning quantity insensitive stress systems via local inference. In *Proceedings of the Eighth Meeting of the ACL Special Interest Group on Computational Phonology and Morphology at HLT-NAACL 2006*, pages 21–30, New York City, USA. Association for Computational Linguistics.

Dileep Kini and Sumit Gulwani. 2015. Flashnormalize: Programming by examples for text normalization. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 776–783. AAAI Press.

Tiago Pimentel, Maria Ryskina, Sabrina J. Mielke, Shijie Wu, Eleanor Chodroff, Brian Leonard, Garrett Nicolai, Yustinus Ghanggo Ate, Salam Khalifa, Nizar Habash, Charbel El-Khaissi, Omer Goldman, Michael Gasser, William Lane, Matt Coler, Arturo Oncevay, Jaime Rafael Montoya Samame, Gema Celeste Silva Villegas, Adam Ek, Jean-Philippe Bernardy, Andrey Shcherbakov, Aziyana Bayyr-ool, Karina Sheifer, Sofya Ganieva, Matvey Plugaryov, Elena Klyachko, Ali Salehi, Andrew Krizhanovsky, Natalia Krizhanovsky, Clara Vania, Sardana Ivanova, Aelita Salchak, Christopher Straughn, Zoey Liu, Jonathan North Washington, Duygu Ataman, Witold Kieraś, Marcin Woliński, Totok Suhardijanto, Niklas Stoehr, Zahroh Nuriah, Shyam Ratan, Francis M. Tyers, Edoardo M. Ponti, Grant Aiton, Richard J. Hatcher, Emily Prud'hommeaux, Ritesh Kumar, Mans Hulden, Botond Barta, Dorina Lakatos, Gábor Szolnok, Judit Ács, Mohit Raj, David Yarowsky, Ryan Cotterell, Ben Ambridge, and Ekaterina Vylomova. 2021. Sigmorphon 2021 shared task on morphological reinflection: Generalization across languages. In *Proceedings of the 18th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 229–259, Online. Association for Computational Linguistics.

Oleksandr Polozov and Sumit Gulwani. 2015. Flashmeta: A framework for inductive program synthesis. *SIGPLAN Not.*, 50(10):107–126.

Partho Sarthi, Monojit Choudhury, Arun Iyer, Suresh Parthasarathy, Arjun Radhakrishna, and Sriram Rajamani. 2021. ProLinguist: Program Synthesis for Linguistics and NLP. *IJCAI Workshop on Neuro-Symbolic Natural Language Inference.*

Armando Solar-Lezama. 2008. *Program Synthesis by Sketching*. Ph.D. thesis, University of California, Berkeley.

Saujas Vaduguru, Aalok Sathe, Monojit Choudhury, and Dipti Sharma. 2021. Sample-efficient linguistic generalizations through program synthesis: Experiments with phonology problems. In *Proceedings of the 18th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 60–71, Online. Association for Computational Linguistics.

Shijie Wu. 2020. Neural transducer. `https://github.com/shijie-wu/neural-transducer/`.

Shijie Wu, Edoardo Maria Ponti, and Ryan Cotterell. 2021. Differentiable generative phonology.

Cross-lingual Alignment of Knowledge Graph Triples with Sentences

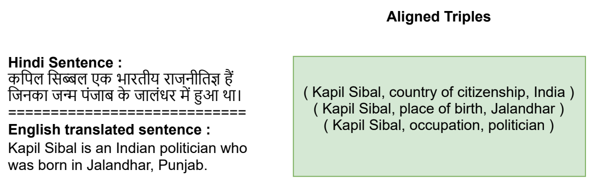
Swayatta Daw[1]*, **Shivprasad Sagare**[2]*, **Tushar Abhishek**[2]*,
Vikram Pudi[1] **and Vasudeva Varma**[2]
[1]Data Sciences and Analytics Center, IIIT Hyderabad, India
[2]Information Retrieval and Extraction Lab, IIIT Hyderabad, India
{swayatta.daw, shivprasad.sagare, tushar.abhishek}@research.iiit.ac.in
{vikram, vv}@iiit.ac.in

Abstract

The pairing of natural language sentences with knowledge graph triples is essential for many downstream tasks like data-to-text generation, facts extraction from sentences (semantic parsing), knowledge graph completion, etc. Most existing methods solve these downstream tasks using neural-based end-to-end approaches that require a large amount of well-aligned training data, which is difficult and expensive to acquire. Recently various unsupervised techniques have been proposed to alleviate this alignment step by automatically pairing the structured data (knowledge graph triples) with textual data. However, these approaches are not well suited for low resource languages that provide two major challenges: (1) unavailability of pair of triples and native text with the same content distribution and (2) limited Natural language Processing (NLP) resources. In this paper, we address the unsupervised pairing of knowledge graph triples with sentences for low resource languages, selecting Hindi as the low resource language. We propose cross-lingual pairing of English triples with Hindi sentences to mitigate the unavailability of content overlap. We propose two novel approaches: NER-based filtering with Semantic Similarity and Key-phrase Extraction with Relevance Ranking. We use our best method to create a collection of 29224 well-aligned English triples and Hindi sentence pairs. Additionally, we have also curated 350 human-annotated golden test datasets for evaluation. We make the code and dataset publicly available [†] and hope that this will help advance further research in this critical area.

1 Introduction

The pairing of structural data (knowledge graphs, Abstract Meaning Representations (AMRs), tables,


Aligned Triples

Hindi Sentence :
कपिल सिब्बल एक भारतीय राजनीतिज्ञ हैं
जिनका जन्म पंजाब के जालंधर में हुआ था।
==============================
English translated sentence :
Kapil Sibal is an Indian politician who
was born in Jalandhar, Punjab.

(Kapil Sibal, country of citizenship, India)
(Kapil Sibal, place of birth, Jalandhar)
(Kapil Sibal, occupation, politician)

Figure 1: A Cross-lingual English triple and Hindi text Example (with English Translation)

databases, etc.) with natural languages sentences has led to the development of many downstream tasks such as Relation extraction (Ji et al., 2017), Knowledge graph population (Vu et al., 2021), dialog generation (Wen et al., 2016), Generation of natural text from structured data (Gardent et al., 2017; Parikh et al., 2020; Mager et al., 2020), etc.

Most existing methods solve above downstream tasks using neural-based end-to-end approaches that require a large amount of well-aligned human-annotated training data. However, the human-annotated dataset is expensive and difficult to obtain as annotators need to understand the structured data and natural text across various domains thoroughly. To overcome the lack of labeled data and difficulty in domain adaptation, unsupervised alignment has recently emerged as an active area of research (Fu et al., 2020; Agarwal et al., 2020; Fan and Gardent, 2020). Most of these unsupervised approaches utilize a large amount of structural and textual data having high content overlap. However, extending these approaches to low resource languages still poses a challenge due to the lack of structured data that has same content distribution as textual data.

In this work, we propose cross-lingual pairing of English triples with native language sentences to mitigate the unavailability of semantic content overlap for low resource languages. We select Hindi as low resource language for evaluating the

[†]https://www.dropbox.com/sh/lrh5q9odadixmqx/AABrTT7YjN6-xVLvviNpqQM6a?dl=0

*Equal Contribution

efficiency of cross-lingual alignment. We explore alignment between the English triples present in Wikidata (Vrandečić and Krötzsch, 2014) with sentences extracted from Hindi Wikipedia articles.

Specifically, through this work, we aim to achieve the following objectives:

1. We introduce solid baselines for the cross-lingual alignment task and propose two novel approaches: NER-based filtering with Semantic Similarity and Key-phrase Extraction with Relevance Ranking. All the approaches mentioned in the paper can be extended to multiple languages, as we do not rely on language-based heuristics.

2. We use our best method to create a collection of 26302 well-aligned English triples and Hindi sentence pairs for training. Similarly, we create validation dataset consisting of 2922 data instances. Additionally, we have also collected 350 human-labeled gold test dataset to evaluate alignment methods.

The remainder of the paper is organized as follows. We discuss related work in Section 2. We discuss the dataset creation details in Section 3. We explain the proposed methods in Section 4. Additionally, we present baselines, experimental settings, results, and analysis in Section 5. Finally, we conclude with a summary of our work and future directions in Section 7.

2 Related work

Recently, there has been a lot of effort in creating automated structured data to text datasets in various domains. (Lebret et al., 2016) introduced a WikiBIO dataset by aligning opening sentences with infoboxes in English Wikipedia articles on person's biographies. Several extensions of this method of aligning Wikipedia text with infoboxes have been proposed to create a dataset in different languages (Nema et al., 2018) and domains (Qader et al., 2018). Datasets created using these methods are constrained to a specific domain. (Fu et al., 2020) alleviates this limitation by aligning knowledge graph triples in Wikidata with opening sentences in Wikipedia. It uses lexical overlap between the name entities present in a sentence, and Wikidata triples for alignment. In addition to using triples available in Wikidata (Wikipedia's Knowledge Graph), (Agarwal et al., 2020) introduced a dataset that also incorporates sub-property information in the form of quadruples. These datasets focus on aligning either knowledge graph triples or infoboxes with sentences present in Wikipedia articles. (Chen et al., 2021) introduced a dataset that combined the structured information residing in Wikidata and infoboxes with a given sentence. To scale alignment of structured data with natural text across various domains (Elsahar et al., 2018; Jin et al., 2020) introduced sequential pipeline strategy consisting of data collection, data filtering, entity linking, and alignment. Additionally, it also suggests incorporating a human-annotated test dataset to evaluate the different alignment methods.

All of the previous approaches depend upon lexical overlap between structured and textual data. These approaches are ineffective for cross-lingual alignment where structured data and textual data are in different languages. Although, we can utilize previously proposed strategies for dataset creation by translating either structured data or textual data to other languages. WebNLG 2020 (Castro Ferreira et al., 2020) shared task presents one such cross-lingual aligned dataset where Shimorina et al. (2019) performs automatic translation and post editing of English sentences to Russian. Final dataset consists of English triples aligned with Russain sentences verbalizing those triples. Such approaches do incur the loss due to automatic translation though. Later, we demonstrate that our proposed approach for cross-lingual alignment achieves comparatively better results.

3 Dataset Creation

3.1 Data collection

We use Wikidata as our Knowledge Graph (KG) for obtaining English triples and Hindi Wikipedia for fetching equivalent sentences. There exists an unambiguous one-to-one mapping between Wikidata entities and Wikipedia articles, which enables us to collect high-quality data for many entities. We initially explored all domains and subdomains of Wikipedia articles. We decided to choose the *person* domain in Hindi Wikipedia as it contains the maximum number of entities within a domain (˜16% of Hindi Wikipedia), allowing us to create a larger dataset. The article text and English triples are fetched and pre-processed for each entity having a Hindi Wikipedia page. Triples with non-useful predicates like external identifiers, URLs, etc., are removed. We extract the first three sentences from each article using sentence tokenization in Hindi. This data acts as the input to our alignment models, which predict a relevant set of triples for each sentence out of that particular entity's entire candidate set of triples. We use our best-proposed approach to create a total of 29224 English triple and sentences pair covering 12429 entities.

3.2 Test Set Annotation

We also collected a human-annotated test set of 460 structured data and text pairs, apart from the unsupervised training and validation set. We sample the 460 instances for annotation from the above-collected data and present them to the user in our specially developed web-based UI. The user can see the sentence and all the candidate triples associated with that entity. Two of the authors independently annotated these instances. The Cohen's Kappa score i.e. inter-annotator agreement for the annotations, was found to be 0.74. Finally, with the help of a language expert, the final test data samples were agreed upon from annotations responses of both the authors. We select 350 data instances as test datasets on which we report the metrics scores of our approaches. The remaining 110 samples are used as internal validation set to tune the hyperparameters like threshold values.

The distribution of sentences and other statistics across different domains can be found in table 1.

Domain	**Entity count**	**Sentence count**	**Sentence count (in test data)**	**Avg sentence length (in test data)**	**Avg fact count (in test data)**
Actors	2106	5469	50	14.32	3.60
Cricketers	2316	4694	100	21.19	4.70
Politicians	3906	8916	100	18.64	3.47
Writers	2755	6629	50	15.65	1.78
Singers	739	1944	25	18.04	2.92
Journalists	607	1572	25	17.32	2.12
Total	12429	29224	350	17.52	3.08

Table 1: Table contains entity count and sentence count for final aligned dataset across different domains. It also presents statistics of manually annotated test data for each domain.

4 Unsupervised Cross-lingual Alignment

Our alignment model aims to align the most relevant English triples to Hindi sentences. We introduce two novel approaches for cross-lingual sentence and facts alignment task: 1) NER-based filtering with Semantic Similarity and 2) Key-phrase Extraction with Relevance Ranking.

NER-based filtering with Semantic Similarity incorporates a novel idea for Named Entity Disambiguation. We used Nearest Neighbor-based Search to find the most relevant English words for the given Hindi words in the sentence by projecting Hindi and English words in the same vector space. We use Multilingual Unsupervised and Supervised Embeddings (MUSE) (Lample et al., 2017) to obtain multilingual vector representation and then perform the Nearest Neighbor Search to obtain the top-k candidates. The chosen candidates are further filtered based on semantic similarity, which boosts the precision of the model. We experiment with several state-of-the-art multilingual transformer-based models to find semantic similarities between facts and sentences.

In Key-phrase Extraction with Relevance Ranking, we extract key phrases from a Hindi sentence based on simple POS-tag-based heuristics and then rank extracted key phrases in the sentence to their relevance with its corresponding constituent article. We propose a new multilingual variant of EmbedRank (Bennani-Smires et al., 2018) to obtain rankings. Top-k relevant triples are then selected based on similarity scores with the key phrases of a sentence.

4.1 NER-based filtering with Semantic Similarity

To obtain matching English triples for a given Hindi sentence s, the idea is to filter the triples using named entity recognition before matching them on semantic similarity. Our assumption is based on the observation that if a triple has a Named Entity, then the sentence with which it aligns will also have the same or a variation of that Named Entity. If a triple does not have a Named Entity, we consider it for finding semantic similarity with the sentence.

We concatenate each word in the triple together and then extract named entities from it. Our goal is to find the overlap between the words in the Hindi sentence to the Named Entities identified in the triple. There can be multiple variations in how a Named Entity is written in an Indian Language such as Hindi. So, using a direct translation would not suffice for the alignment objective. Additionally, there might be translation loss associated with it.

To circumvent this problem, we used a pipeline approach consisting of two stages: 1) Filtering of triples based on bucket approach, and 2) Semantic

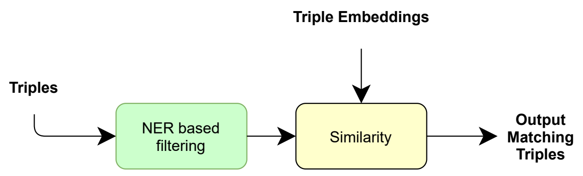


Figure 2: NER-based filtering + Semantic Similarity

similarity approach.

Filtering of triples based on bucket approach creates a bucket of English words by retrieving top-k nearest neighbor English words for each word present in the given Hindi sentence s from the common multilingual vector space created using MUSE (Lample et al., 2017). Then we calculate the intersection of the named entities identified in triple with the previously created bucket of English words for that Hindi sentence s. Finally, we obtain a score for each triple by dividing the amount of intersection by the total number of words present across all the named entities. We retain facts having score above a certain threshold, then proceed with semantic similarity in the next stage.

Semantic similarity approach further refines the triples obtained from the previous stage by calculating the inner product between the Hindi sentence representation and fact representation. Both the sentence level representation and fact level representation are obtained from multilingual transformer models as discussed in Section 5. Finally, we retain triples above a certain threshold (different threshold from the previous stage). We have illustrated the pipeline approach in Figure 2.

4.2 Key-phrase Extraction with Relevance Ranking

We extract the Hindi key phrases from the Hindi Wikipedia article based on simple POS-tag-based heuristics for this method. We define a phrase as a key phrase if it contains at least zero or more Adjectives followed by one or more Nouns. These obtained key phrases are ranked on how semantically similar they are to the input Hindi Wikipedia article. We call this process Key-phrase Extraction with Relevance Ranking. The ranking mechanism follows a multilingual variant of the EmbedRank (Bennani-Smires et al., 2018) method. The intuition behind EmbedRank is to embed candidate phrases and the corresponding article in the same high-dimensional vector space. Then, the key phrases are ranked based on closeness with

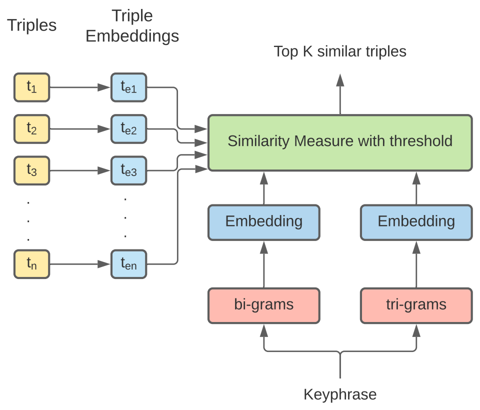


Figure 3: Method to return Top K triples from key phrases

the article in the same vector space. Our variant is explained in Algorithm 1.

Algorithm 1: Ranking key phrases with respect to Article Relevance

```
1. Let $N$ = { set of all key phrases in article
   $A$}.
2. Concatenate all the key phrases in $N$ and
   let $Nv \leftarrow$ vector representation of the
   concatenated key phrases.
3. For a sentence $s$ in the article $A$, $M \leftarrow$
   set of all extracted key phrases from $s$. So,
   $M \subseteq N$.
4. For each key phrase $K$ in $M$, let $Kv \leftarrow$
   vector representation of $K$.
5. Assign a $score$ to $K$ , where $score$ =
   similarity between $Kv$ and $Nv$.
6. Rank all the key phrases in $M$ based on
   the $score$.
```

The process of obtaining similar triples from ranked key phrases is explained in Figure 3. After the key phrases are ranked for an article A, we extract n-grams for each key phrase. We find the vector embeddings for each n-gram and each triple. Now, each n-gram is compared with each triple, and a semantic similarity score is obtained. We keep the best matching triple for each n-gram. Then, we obtain the most similar triples per n-gram for a key phrase. Among them, we select top-k triples. These top-k triples form the most relevant triples for a key phrase. We combine the results from all key phrases in a Hindi sentence to obtain sentence-level matches.

5 Experiments

5.1 Baselines

We experimented with the following baselines:
Multilingual Universal Sentence Encoder (Yang et al., 2019a) is a general-purpose sentence embedding model for transfer learning and semantic text retrieval tasks. It relies on a standard dual-encoder neural framework with shared weights, trained in a multi-task setting with an additional translation bridging task. We use the same strategy to filter out the fact triples as mentioned for mBERT.
Word Overlap uses K Nearest neighbor search to choose K-most relevant English words for each Hindi word present in the Hindi sentence. The word search happens in a multilingual vector space created using MUSE (Lample et al., 2017). We keep all these top-K English words in a bucket. Then, we calculate the overlap between words in the triple and the English words in the bucket for each sentence. If the overlap is above a certain threshold, we classify that triple as aligned with that sentence.
Static Sentence Similarity use MUSE (Lample et al., 2017) to obtain multilingual word embeddings. We find the average of these word embeddings to create sentence representation for a Hindi sentence. We average all the word embeddings in a triple to obtain fact-level representation for that triple. Finally, we find the cosine similarity between sentence level and fact level representation and retain triples above a certain threshold for a given Hindi sentence.
mBERT (Devlin et al., 2018) (multilingual Bidirectional Encoder Representations from Transformers) encodes both the Hindi sentence and list of associated facts. Facts are verbalized by concatenating the subject, predicate, and object. We obtain the vector representations by taking the average of sub-word representation from the last layer of mBERT (mean pooling). Then, we find the cosine similarity score between the sentence and fact-level representation. Finally, we retain fact triples whose similarity score is greater than a certain threshold.
MuRIL (Khanuja et al., 2021) (Multilingual Representations for Indian Languages) is pre-trained on a significantly large amount of Indian text corpora with an extensive vocabulary for Indian languages. With MuRIL, we use the same strategy to filter out the fact triples as mentioned for mBERT.
LaBSE (Feng et al., 2020)(Language-Agnostic BERT Sentence Embedding) is a multilingual embedding model that encodes text from different languages into a shared embedding space pre-trained using the Masked Language Modeling and Translation Language Modeling objectives. With LaBSE, we use the same strategy to filter out the fact triples as mentioned for mBERT.
XLM-R (STS) and XLM-R (Paraphrase) are sentence transformers that fine-tune XLM-Roberta (Conneau et al., 2019) on semantic text similarity (STS) (Cer et al., 2017) and on multilingual paraphrase dataset (Yang et al., 2019b) respectively.

5.2 Experimental Settings

For the Word Overlap approach, we set the threshold value to 1 and fixed k=5 in k nearest neighbor retrieval. We translate the words which are out of the vocabulary. For all multilingual transformer-based methods: mBERT, MuRIL, LaBSE, multilingual universal sentence encoder, XLM-R, we use the base model available (consists 12 layers) on Huggingface (Wolf et al., 2020).

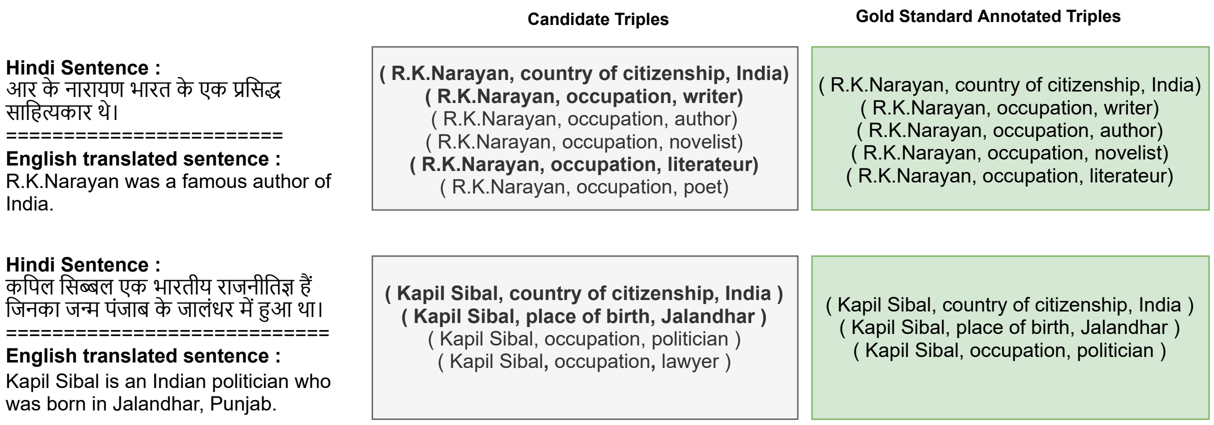
Threshold value	F1-Score
0.35	0.48
0.45	**0.55**
0.55	0.52
0.65	0.38

Table 2: Threshold values for sentence-triple semantic similarity on internal validation set for XLM-R (base)

The threshold value is set to 0.45 for cosine similarity after hyperparameter tuning on our internal validation dataset. We tried various pooling strategies like [CLS] token representation, sum pooling, and mean pooling for sentence-level representation. We found that mean pooling consistently performs the best.

K	F1-Score
3	0.65
4	0.72
5	**0.74**
6	0.66
7	0.67
8	0.68
9	0.66
10	0.63

Table 3: K value for K-Nearest neighbor for NER-based filtering with Semantic Similarity method (tested on internal validation set)



	Candidate Triples	Gold Standard Annotated Triples
Hindi Sentence : आर के नारायण भारत के एक प्रसिद्ध साहित्यकार थे। ======================== **English translated sentence :** R.K.Narayan was a famous author of India.	**(R.K.Narayan, country of citizenship, India)** **(R.K.Narayan, occupation, writer)** (R.K.Narayan, occupation, author) (R.K.Narayan, occupation, novelist) **(R.K.Narayan, occupation, literateur)** (R.K.Narayan, occupation, poet)	(R.K.Narayan, country of citizenship, India) (R.K.Narayan, occupation, writer) (R.K.Narayan, occupation, author) (R.K.Narayan, occupation, novelist) (R.K.Narayan, occupation, literateur)
Hindi Sentence : कपिल सिब्बल एक भारतीय राजनीतिज्ञ हैं जिनका जन्म पंजाब के जालंधर में हुआ था। ============================ **English translated sentence :** Kapil Sibal is an Indian politician who was born in Jalandhar, Punjab.	**(Kapil Sibal, country of citizenship, India)** **(Kapil Sibal, place of birth, Jalandhar)** (Kapil Sibal, occupation, politician) (Kapil Sibal, occupation, lawyer)	(Kapil Sibal, country of citizenship, India) (Kapil Sibal, place of birth, Jalandhar) (Kapil Sibal, occupation, politician)

Figure 4: The first example is a predicted sample from the Key-phrase Extraction with Relevance Ranking approach. The second example is a predicted sample for the NER based filtering with Semantic Similarity approach. The prediction by each model is highlighted in bold in the candidate triples.

We determine the optimal K for K-Nearest neighbors and the optimal similarity threshold by tuning these hyperparameters on the internal validation set consists of 110 instances. We provide the detailed results of this hyperparameter search in Table 2 and Table 3. We obtain the optimal value for K in K-Nearest Neighbors as 5. Similarly, we observe the optimal value for the similarity threshold to be 0.45. We use XLM-R (base) as the reference transformer-based model as it is the best performing baseline.

For recognizing named entities, we use a BERT-CRF tagger trained on the OntoNotes dataset (Weischedel et al., 2017). We use AllenNLP (Gardner et al., 2017) for our NER implementation.

For Key-phrase Extraction with Relevance Ranking, we set n-gram values $\in [2, 3]$ and use Stanford coreNLP (Manning et al., 2014) to detect POS-tags. We used XLM-R (Paraphrase) as the multilingual transformer encoder with a similarity threshold of 0.45.

5.3 Evaluation Metric and Results

We use micro-average *Precision*, *Recall* and *F1-Score* as our evaluation metrics.From the results in Table 4, it is evident that MuRIL performs better than mBERT as it is solely pre-trained on Indian languages with extensive vocabulary size. Surprisingly, a simple approach like word overlap has higher recall than MuRIL. The reason is that it searches k-nearest neighbors in a multilingual vector space, as explained in section 5.1. So, this process captures more word variations while retrieving the facts. XLM-R (paraphrase) model in baselines performs better than other multilingual transformers as it is fine-tuned on the downstream tasks specific to text similarity. LaBSE is pre-trained on the translation language modeling loss. So, it effectively captures the semantic similarity between facts and sentences of different languages.

We observe that Key-phrase Extraction with Relevance Ranking has high precision. As the process captures the relevance of each key phrase with its article, it ensures to keep only those key phrases that are highly relevant to the article. The matches are refined further by n-gram matching with triples.

Surprisingly, NER-based filtering with Semantic Similarity gives the highest performance in terms of both precision and recall. This result shows that the most relevant fact triples are significantly biased towards having named entities as the primary factual information. Therefore, even though our Key-phrase Ranking method considers the entire context of an article to obtain relevant phrases, the NER-based model still performs better.

6 Error Analysis

Key-phrase Extraction with Relevance Ranking: As per the ranking mechanism, we keep only the most relevant top ranked triples. However, we notice that in some cases, especially where there are multiple triples which convey similar information, the model misses to capture all the relevant triples. Only the highest rank triples are considered, which leads to similar triples being missed out due to a slightly lower rank. In Figure 4, the first example is a predicted sample by the Key-phrase extraction model. We observe that occupation:author and oc-

S.no	Approaches	Precision	Recall	F1-Score
1	mBERT (mean pooling)	0.37	0.31	0.33
2	Static Sentence Similarity	0.38	0.48	0.42
3	Multilingual Universal Sentence Encoder	0.62	0.38	0.47
4	Word Overlap	0.50	0.52	0.51
5	LaBSE (mean pooling)	0.49	0.56	0.52
6	XLM-R (STS)	0.57	0.48	0.52
7	MuRIL (mean pooling)	0.55	0.51	0.53
8	XLM-R (paraphrase)	0.52	0.58	0.55
9	Key-phrase Extraction with Relevance Ranking	0.78	0.72	0.75
10	NER based filtering with Semantic similarity	**0.79**	**0.83**	**0.81**

Table 4: Precision, Recall and F1-score across different approaches.

cupation:novelist are missed out by the model, due to the ranking mechanism.

NER based filtering with Semantic Similarity: We notice that sometimes fact triples without named entities are being missed by the model. The second example in Figure 4 is a predicted sample by the NER-based model. We observe that occupation: politician has been ignored by the model, as ”politician” is not a named entity.

7 Conclusion

We investigate the unexplored problem of cross-lingual alignment of English triples with sentences for low-resource languages like Hindi. This paper demonstrates the result over several baselines ranging from simple techniques like word overlap to more complex approaches that use pre-trained language models. Finally, we propose two novel methods of NER-based filtering with Semantic Similarity and Key-phrase Extraction with Relevance Ranking. We show through our experiments that these approaches perform better than the baselines on the human-annotated gold dataset, which we have created as a part of this project. We created a large dataset of English triples mapped with Hindi sentences using our best-performing model, making it publicly available for further research.

We plan to use the cross-lingual aligned dataset for various NLP tasks like text generation, KB population, and concept extraction for future work. Also, we are planning to extend this work to other Indian languages. We strongly believe that our alignment models and dataset will enhance the research undertaken for low-resource languages in the scientific community.

References

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2020. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. *arXiv preprint arXiv:2010.12688*.

Kamil Bennani-Smires, Claudiu Musat, Andreea Hossmann, Michael Baeriswyl, and Martin Jaggi. 2018. Simple unsupervised keyphrase extraction using sentence embeddings. *arXiv preprint arXiv:1801.04470*.

Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina. 2020. The 2020 bilingual, bi-directional WebNLG+ shared task: Overview and evaluation results (WebNLG+ 2020). In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 55–76, Dublin, Ireland (Virtual). Association for Computational Linguistics.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Mingda Chen, Sam Wiseman, and Kevin Gimpel. 2021. Wikitablet: A large-scale data-to-text dataset for generating wikipedia article sections. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 193–209.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

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*.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018. T-rex: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Angela Fan and Claire Gardent. 2020. Multilingual amr-to-text generation. *arXiv preprint arXiv:2011.05443*.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2020. Language-agnostic bert sentence embedding. *arXiv preprint arXiv:2007.01852*.

Zihao Fu, Bei Shi, Wai Lam, Lidong Bing, and Zhiyuan Liu. 2020. Partially-aligned data-to-text generation with distant supervision. *arXiv preprint arXiv:2010.01268*.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188, Vancouver, Canada. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Zhijing Jin, Qipeng Guo, Xipeng Qiu, and Zheng Zhang. 2020. Genwiki: A dataset of 1.3 million content-sharing text and graphs for unsupervised graph-to-text generation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2398–2409.

Simran Khanuja, Diksha Bansal, Sarvesh Mehtani, Savya Khosla, Atreyee Dey, Balaji Gopalan, Dilip Kumar Margam, Pooja Aggarwal, Rajiv Teja Nagipogu, Shachi Dave, et al. 2021. Muril: Multilingual representations for indian languages. *arXiv preprint arXiv:2103.10730*.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. *arXiv preprint arXiv:1603.07771*.

Manuel Mager, Ramón Fernandez Astudillo, Tahira Naseem, Md Arafat Sultan, Young-Suk Lee, Radu Florian, and Salim Roukos. 2020. GPT-too: A language-model-first approach for AMR-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1846–1852, Online. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Preksha Nema, Shreyas Shetty, Parag Jain, Anirban Laha, Karthik Sankaranarayanan, and Mitesh M Khapra. 2018. Generating descriptions from structured data using a bifocal attention mechanism and gated orthogonalization. *arXiv preprint arXiv:1804.07789*.

Ankur P Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. Totto: A controlled table-to-text generation dataset. *arXiv preprint arXiv:2004.14373*.

Raheel Qader, Khoder Jneid, François Portet, and Cyril Labbé. 2018. Generation of company descriptions using concept-to-text and text-to-text deep models: dataset collection and systems evaluation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 254–263. Association for Computational Linguistics.

Anastasia Shimorina, Elena Khasanova, and Claire Gardent. 2019. Creating a corpus for Russian data-to-text generation using neural machine translation and post-editing. In *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, pages 44–49, Florence, Italy. Association for Computational Linguistics.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Binh Vu, Craig A Knoblock, Pedro Szekely, Minh Pham, and Jay Pujara. 2021. A graph-based approach for inferring semantic descriptions of wikipedia tables. In *International Semantic Web Conference*, pages 304–320. Springer.

Ralph M. Weischedel, Eduard H. Hovy, Mitchell P. Marcus, and Martha Palmer. 2017. Ontonotes : A large training corpus for enhanced processing.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016. Multi-domain neural network language generation for spoken dialogue systems. *arXiv preprint arXiv:1603.01232*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yinfei Yang, Daniel Cer, Amin Ahmad, Mandy Guo, Jax Law, Noah Constant, Gustavo Hernandez Abrego, Steve Yuan, Chris Tar, Yun-Hsuan Sung, et al. 2019a. Multilingual universal sentence encoder for semantic retrieval. *arXiv preprint arXiv:1907.04307*.

Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. 2019b. PAWS-X: A Cross-lingual Adversarial Dataset for Paraphrase Identification. In *Proc. of EMNLP*.

Introduction to ProverbNet: An Online Multilingual Database of Proverbs and Comprehensive Metadata

Shreyas Pimpalgaonkar, Dhanashree Lele,
Malhar Kulkarni, and Pushpak Bhattacharyya
Indian Institute of Technology Bombay, India
`shreyas.pgkr@gmail.com, dhanashreelele01@gmail.com`
`malhar@hss.iitb.ac.in, pb@cse.iitb.ac.in`

Abstract

Proverbs are unique linguistic expressions used by humans in the process of communication. They are frozen expressions and have the capacity to convey deep semantic aspects of a given language. This paper describes ProverbNet, a novel online multilingual database of proverbs and comprehensive metadata equipped with a multipurpose search engine to store, explore, understand, classify and analyse proverbs and their metadata. ProverbNet has immense applications including machine translation, cognitive studies and learning tools. We have 2320 Sanskrit Proverbs and 1136 Marathi proverbs and their metadata in ProverbNet and are adding more proverbs in different languages to the network.

1 Introduction

According to Norrick (2015), proverbs are self-contained, didactic linguistic units, and their content is usually fixed and poetic. Proverbs contain the same words as ordinary sentences, but are frozen expressions and have a non-compositional meaning, created by humans and handed down through generations (Meider, 2004). Proverbs can be used to present facts, refer to human socio-cultural events, give advice, and critique behavior (Lubis, 2019). Understanding of metaphorical ideas and a priori knowledge is essential for getting from the literal meaning of the proverb to its hidden and intended meaning. The crucial task of machine translation fails to translate proverbs because of the inability to detect proverbs automatically and incorrectness of direct translation. As proverbs are just like any other sentences, it is difficult to detect them in textual corpus automatically. There are a few proverbs that have counterparts in other languages, and sometimes we can obtain them via direct translation, but many times, they are entirely different. Proverbs can be categorized into various classes and categories (Lauhakangas, 2001). They are usually static multi-word expressions having specific keywords. In most cases, we cannot change even a single word in the proverb. If done, then it becomes a simple sentence without an accepted non-compositional meaning.

For all languages in the Indo-European family in India, Sanskrit is considered to be linked to them historically. Sanskrit is one of the oldest documented members of the Indo European family (Woodard, 2008). Since it is an ancient language, the wisdom embedded in Sanskrit proverbs has been carried forward for thousands of years. However, nowadays the use of proverbs has reduced even in the native spoken languages, let alone the use of proverbs in the Sanskrit language. To preserve these wisdom-packed proverbs, we require a digital database with the analysis of the proverbs. Some compilations are found having internet base. However, the collections of Sanskrit proverbs found on the internet are comparatively small and limited only to the famous proverbs. People refer to these compilations when they are in want of a particular proverb, or the proverbs related to a particular theme or concept. For such reference, a mere compilation does not suffice. There should be an online database in a searchable form. As far as Indian languages are concerned, there are no databases of proverbs available and our tool removes this desideratum, and this ProverbNet may act as a single reference point.

2 Literature Survey

To the best of our knowledge, no online multilingual database of proverbs with comprehensive metadata focused on Sanskrit and other Indian Languages exists. We, therefore present a survey of related literature – the sources from which we compiled proverbs, proverbial analysis, cognition, multi-word expressions, wordnets, online proverb databases and possible applications.

There are many offline collections of proverbs. Ābhāṇakajagannātha by Jagannath (Jagannath, 2009) is a compilation of new proverbs in Sanskrit. Manwaring (1899) has collected, translated and classified a total of 1910 Marathi proverbs. Bharatiya Kahavat Sangraha by Naravane (1978) is a collection of proverbs in three volumes of proverbs in fifteen Indian Languages and can be called as the Gītā of Indian Proverbs. Bhosale (2016) has composed a collection of proverbs from Marathi language which may go into oblivion soon and has tried to preserve these proverbs. Such collections are essential in preserving the treasure of language.

A vast amount of literature exists on proverbial analysis. Several books (Honeck and Temple, 1994; Mieder et al., 1994; Honeck, 2016; Bhagwat, 1985; Brough, 1953) and articles (Bronkhorst, 2005; Temple, 1999; Swinney, 1979) provide deep insights into the journey of the meaning of a proverb from literal meaning to its suggestive meaning. Meider (2004) explores definitions, classification, origin, dissemination, collection, and various case studies of proverbs. Grzybek (2014) presents a semiotical study of proverbs in pragmatical, syntactical, and semantical dimensions. A psychological study suggests that the ability or inability to explain meanings of proverbs indicates the presence or absence of abstract thinking abilities, and a strong understanding of proverbs reveals essential insights into how people conceptualise metaphorical ideas (Gibbs and Beitel, 1995). According to Ferretti et al. (2007), people should be able to understand the non-compositional meanings of proverbs directly without having to first think about and reject their literal meanings.

Proverbs belong to a class of multi-word expressions. Therefore, research conducted for the identification and analysis of multi-word expressions can be applied to the analysis of proverbs. Tsvetkov and Wintner (2014) present a survey on analysing multi-word expressions and provides a more in-depth understanding of tasks related to multi-word expressions like discovery, identification and translation.

When it comes to creating and populating proverb databases, automatic identification of proverbs from a textual corpus is beneficial. Quite a few ways of automatically identifying proverbs exist. Rassi et al. (2014) identify proverbs and their variants by creating a finite state automaton with thirteen commonly found syntactic patterns in proverbs and then test on Brazilian Portuguese journalistic corpus. Sidhu et al. (2010) describes a word-based algorithm that splits input text and known proverbs and to find proverb variants in a given text and uses this to perform machine translation. Garg and Goyal (2014) perform automatic extraction of idioms, proverbs and their variations from text using statistical approaches.

A significant amount of research has been done in the creation of lexical databases. Princeton WordNet (G. A. Miller, 1990) is a massive database of English lexicons. Sanskrit wordnet (Kulkarni et al., 2010) is more than just a dictionary and gives different relations between synsets which represent unique concepts. Indo wordnet (Bhattacharyya, 2010) is a wordnet in multiple Indian Languages. These wordnets are extensive, but their focus is on single words and not multi-word expressions or proverbs.

Online proverb databases usually attempt at aggregating proverbs of one or two languages. Lauhakangas (2013) created a multilingual database of proverbs in European Languages. Our database is focused towards Indian languages, contains more metadata, flexible categorisation, more features and addresses a different set of goals.

3 Main Contribution

We introduce ProverbNet, an intricate network of multilingual proverbs equipped with a multipurpose search engine that contains extensive metadata and detailed cognitive analy-

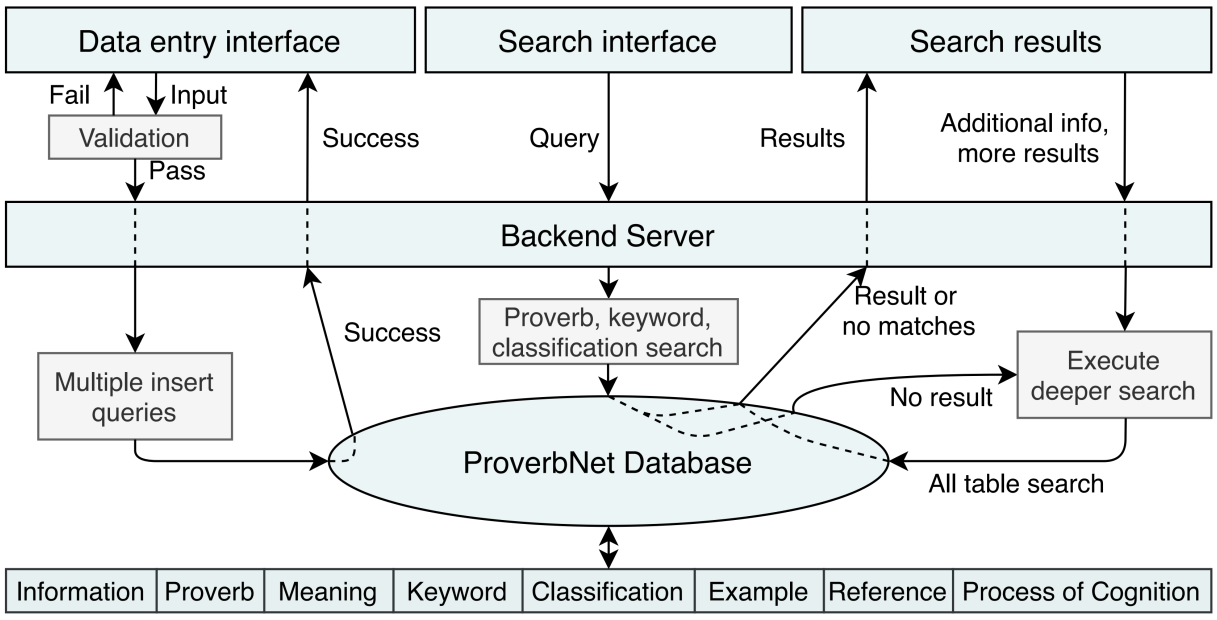


Figure 1: ProverbNet Block Diagram

sis. We gather inspiration from Sanskrit WordNet (Kulkarni et al., 2010) and Indo WordNet (Bhattacharyya, 2010) to design the interface. In order to populate this database and ameliorate the data entry process, we have designed an easy to use and elegant data entry interface. We have also designed a search engine to perform queries on this database. The block diagram of the system is shown in figure 1. We will now explain the system, and describe the details of various fields and their importance in the next section.

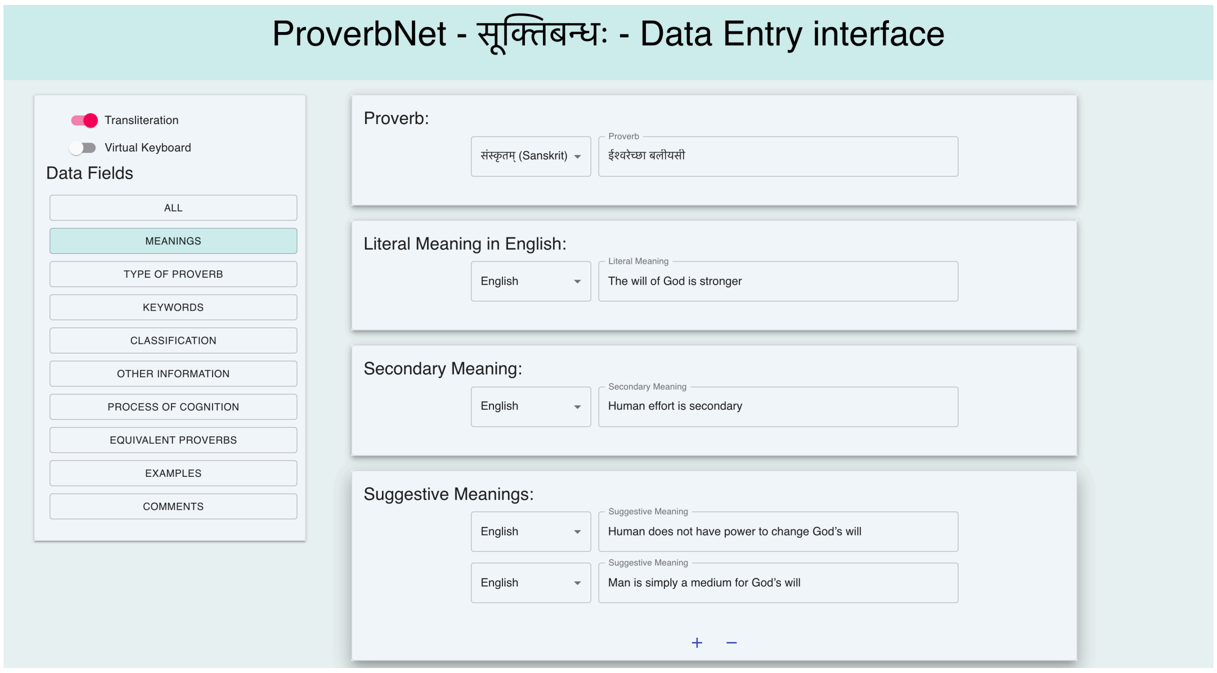
4 ProverbNet Database and Data Entry Interface

We have designed ProverbNet to contain comprehensive metadata about proverbs. We choose and present the data such that it benefits both, an amateur reader and a proficient researcher in Paremiology. The data entry interface (figure 2) is fast, responsive, and elegant. The interface is written in React, a JavaScript framework, and uses material-UI components. Currently, we have 2320 Sanskrit proverbs in the ProverbNet database. Amongst them, 1000 proverbs frequently occur in typical Sanskrit dialogues. Rest of the proverbs are from Ābhāṇakajagannātha, a book that comprises of 1320 new and rare Sanskrit proverbs. We also have 1366 Marathi proverbs, and we are adding proverbs of other languages to the database.

The Data Entry Interface enhances the data entry experience by providing various features like language selectors for appropriate data fields, phonetic transliteration from English to any of the selected languages, and a virtual keyboard to type in the desired language. When the phonetic transliteration option is selected, and a language for a particular text box is chosen, one can type a word in English and press space to get the phonetic transliteration of the word into that language. If the transliterated word is incorrect, the user can press backspace to get more word recommendations, and they can replace their word accordingly. A virtual keyboard of the selected language appears under each text field when the option is enabled. The interface has different types of input fields depending on the type of data. On clicking the submit button, the data gets validated and then gets entered into the database. Now we explain the importance of the fields present in our database and the data entry interface.

Proverb: The user can enter a proverb in their language of choice.

Literal Meaning: The literal meaning of the proverb, preferably in English. It is what someone would say if they considered it as a typical sentence without any figurative meaning.


ProverbNet - सूक्तिबन्धः - Data Entry interface

Transliteration
Virtual Keyboard

Data Fields

Proverb:
संस्कृतम् (Sanskrit)
Proverb: ईश्वरेच्छा बलीयसी

Literal Meaning in English:
English
Literal Meaning: The will of God is stronger

Secondary Meaning:
English
Secondary Meaning: Human effort is secondary

Suggestive Meanings:
English
Suggestive Meaning: Human does not have power to change God's will
English
Suggestive Meaning: Man is simply a medium for God's will

Figure 2: Data entry interface

Secondary meanings: Most of the times, the literal meaning of a proverb does not disclose the actual meaning it wants to convey. Hence, after following the literal meaning of the proverb, we have to advance to its secondary meaning, lakṣaṇā.

Suggestive meanings: A proverb is not uttered in vacuum or isolation. A proverb is almost always a reactionary utterance to either refer to something that has happened or to refer to something that will probably take place. So knowing the context in which we use the proverb becomes vital in understanding the essence of a proverb. The same proverb can be uttered in multiple situations by different speakers. This subjectivity is not covered in secondary meaning (Lakṣaṇā) as secondary meaning limits itself to figurative meaning.Hence a third vṛtti is required to reach the essence of a proverb. This third vṛtti is termed vyañjanā, [the suggestive power (meaning)] in Sanskrit linguistics. The metaphorical meaning takes into consideration how a proverb yields itself to multiple situations. Understanding, the literal and figurative meanings of a proverb, is only a part of comprehending the proverb completely. The comprehension of a proverb as a whole (and not just the meaning) will be complete only when the listener can relate the proverb to different situations. Therefore the suggestive meaning expresses the essence of the proverb. Literal, figurative, and suggestive meanings (i.e., Abhidhā, Lakṣaṇā, Vyañjanā) of a proverb are present in the ProverbNet. Proverbs are 'frozen expressions' with universally accepted fixed meanings hence the variations with reference to context will be minimum.

Cognition of the proverb: Most proverbs may allude to some universal truths or common human behaviours. Thus like morale of a story, cognition of proverb gives morale of the proverb wherever applicable.

Keywords and keyword categories: Lauhakangas (2001) describes classification of proverbs into different categories. For better clustering of proverbs and search, we enter certain keywords of those proverbs and define categories. We select the important words from the proverb or their synonyms as keywords. Further, we classify these keywords into emotions, animal names, body parts, and similar categories. This facilitates the search of proverbs containing specific words.

Type of proverb: The nature of proverbs are classified into five broad categories, viz. observation, criticism, comment, suggestion, advice. Based on the type of the proverb, the user may select zero or more of these types.

Observations: The user may enter any specific observations regarding the use of the proverb. Sometimes proverb and its literal meaning may lead to a positive outcome, but the action is aimed at the elimination of harmful things. For example, Mule Kuthār | The English translation of the proverb is 'Nipping in the bud', which means destroying a thing from its root. While we normally perceive destruction as bad, if the outcome is positive, the action may still be termed as good. Since the proverb suggests destruction, the observation for this proverb is 'negative polarity'. Another example of negative polarity is Vināshkale Viparitbuddhi | Here the words have a negative tone, and the outcome is also negative. On the other hand, carāti carato bhaga:| is a proverb having neither a negative tone nor a negative outcome. The observation for this proverb is 'positive polarity'.

Examples: The user may enter examples of the proverb in a contextual paragraph. The understanding of the use of proverb and its nuances grows with repeated exposure to a proverb. Therefore, examples would help the reader understand the use of proverb and show a possible usage of the proverb as well.

Reference: The user may enter the source of a proverb if available. In Sanskrit, many proverbs originate from Sanskrit Subhaṣhitas. Many Subhaṣhitas have a structure where the first line or first three lines provide an example, and the last line summarises the observation. The last line is commonly used as proverb rather than quoting entire shloka. Many of us are not even aware that a particular *sūkti* is part of a verse. If we understand the whole verse, it becomes easier to understand the essence of that *sūkti* on the backdrop of the thought/ idea conveyed in the whole verse. E.g. Maunm sarvārthsadhanm | keeping mum or remaining silent is beneficial. It is the last caraṇa, i.e. line of the following verse.

Ātmno mukhdosheṇ badhyante shuksārikāh |
Bakāstatra na badhyante Maunm sarvārthsadhanm ||

Parrots are caged because of their quality of imitating the human language in a sweet voice, but cranes are not caged because they do not speak. Therefore remaining silent is always beneficial (keeps you free). When we understand this *sūkti* on the backdrop of parrots being caged and the cranes remaining free, we experience the specific shade of this proverb. The same is with a philosophical sūkti which has been drawn on the Bhagavadgītā, Upaniṣhad, Bhāgavatapurāṇa. Thus taking into consideration the background or the source of the *sūkti* helps us in extracting scent of the meaning from the proverb.

For some proverbs, there is reference to a story. Knowledge of the story is essential for complete cognition. A case from English language is 'David v/s Goliath'. While a reader may comprehend it as a tussle between two persons, if the story of David and Goliath is known, one understands it as a fight of a small person with a mightier one. It is better to be aware of the story behind the shloka to comprehend the proverb completely.

For example, the complete shloka of the proverb Fatātopo Bhayankarh | is

Nirvisheṇāpi sarpeṇ kartavyā mahati fanā |
Viṣamstu na vāpyastu fatātopo Bhayankarh ||

However, even understanding the śloka does not complete the comprehension until the story is told. Thus, the proverb may become part of the regular vocabulary, but the śloka may not. Here, we have given the complete subhāṣita/śloka and its source wherever available. Related context and references are also provided where necessary. This helps in better cognition of the proverb.

Parallels in other languages: One can enter proverbs that have equivalent or similar meanings in other languages. Historically, proverbs from Sanskrit can be said to have come down into modern Indian languages sometimes with variations. It is also observed that proverbs in different languages express a similar concept. Such similar proverbs may help a reader better understand the nuanced meaning of a proverb in Sanskrit. It will also throw light on how a proverb may have undergone change when it moved from Sanskrit to other language. An example is illustrated in figure 5. There is already an existing book of parallel proverbs in Indian languages spanning more than 1000 proverbs across around 10 Indian languages including Sanskrit (Naravane, 1978). We will be using it as a resource for building the database. However there is no metadata i.e., it is not searchable for similar proverbs, opposite proverbs, keyword search etc. Along with these features the ProverbNet explains primary, secondary and suggestive meanings of the proverbs and it also contains the examples of the usage.

Specifics: Wherever applicable, the user can add specific references to Mythology, History or Culture. For example, Ishvarechā baliyasee | is related to Indian mythology due to reference to God Shankara. The proverb naro vā kuñjaro vā is related to history as it has reference to a story from Mahābhārata and we can say that the usage of this proverb has started after Mahābhārata. The proverb vasudhaiva kuṭumbakam has specific reference to Indian culture as this thought encompasses the essence of Indian philosophy.

Process of cognition through the process of Shābdabodha: The following kārikā tells us the process of Shābdabodha.

Padjnyānm hi karaṇm dvāram tartar padārthdhi |
Shābdabodh falam tatra vṛttidhi sahakāriṇi ||

It means: (padajñāna) cognition of the word (maybe the sound or the alphabets) is an instrumental cause (the most effective means in Shābdabodh). Cognition of the meaning of the word (padārthdhi) is the gateway of the śābdabodha, vṛtti helps us to lead to the final goal that is śābdabodha, and the result is the verbal cognition. In order to cognise a proverb through the theory of śābdabodha, we analyse proverbs through each of these aspects. This theory is not only limited to Sanskrit, and we can apply this theory to other languages as well. Following are the steps in the process of cognition.

- Padajñāna - cognition of words of the sentence
- Padārthjnyānm – After identifying the words of the sentence by their sounds or by the sequence of alphabets, we understand their meanings.
- vṛtti - (ṣakti, lakṣaṇā, vyanjanā) to understand the meanings of the word we have to rely on the ṣakti or the primary meaning or the denotative power. However, when meanings are incompatible, we have to go to the secondary meaning that is lakṣaṇā. Vyañjanā is the suggestive meaning of the sentence. The literal meaning of an utterance is only a part of its whole meaning. The suggestive power helps us to go beyond the primary and the secondary meaning.
- Saktigraha - We consider different ways of ṣaktigraha to understand the ṣakti.
- Akaṇkṣā, Yogyatā, Sannidhi - These are three conditions, meaning mutual expectancy of words, compatibility of words and proximity of words respectively. When these three conditions are fulfilled, cognition of the text is done successfully.

If a proverb yields to all these aspects, then we say that the proverb is cognisable through the theory of Śābdabodha. We illustrate this process in detail with the help of an example in Appendix A.

Classification: There are various ways to classify proverbs. Classification of proverbs by any criteria is a complicated question, and no comprehensive and standardized solution exists. Each type of classification has its validity, its practical uses, and also its drawbacks. Kuusi (1972), a Professor of Finnish and Comparative Folk Poetry Studies and paremiologist found three aspects, i.e. the idea, structure, and basic core necessary for the classification of proverbs. He proposed that the proverbs with a common idea would be called synonymous proverbs, those with the common scheme would be called proverbs with shared structure and proverbs with images and phrases having the same meaning would be called proverbs with a common basic core. We use and extend his classification to adapt to Sanskrit and other languages. We classify proverbs in the following broad categories. Subcategories will be explained in Appendix B.

1. Proverb Structure: A proverb is an expression or a comment on a particular situation. The lesser the words, the stronger the essence. Thus the structure of the proverb has a great significance in conveying the expression effectively.

2. Proverbial Formulae: All languages possess certain structural formulae that exhibit a high degree of peculiarity towards the proverb as a linguistic form (Coinnigh, 2015). In Sanskrit proverbs, the following proverbial formulae are widely found: Itah.. tath, yatra.. tatra, yathā.. tathā,

yādṛṣm.. tādṛṣm, x vā.. y vā, x vā.. y na vā, na x na y, yah/yā.. sah/sā

3. Themes: Proverbs have a situational significance, and hence the circumstances for which we use proverbs show a wide range of themes. Therefore, theme becomes an essential basis for classification of proverbs. In 'International type system of proverbs', Matti Kussi provides a framework of specific themes for classificational international proverbs. Using Matti Kussi's framework as a base, this ProverbNet will explore and contribute to the thematic classification of proverbs

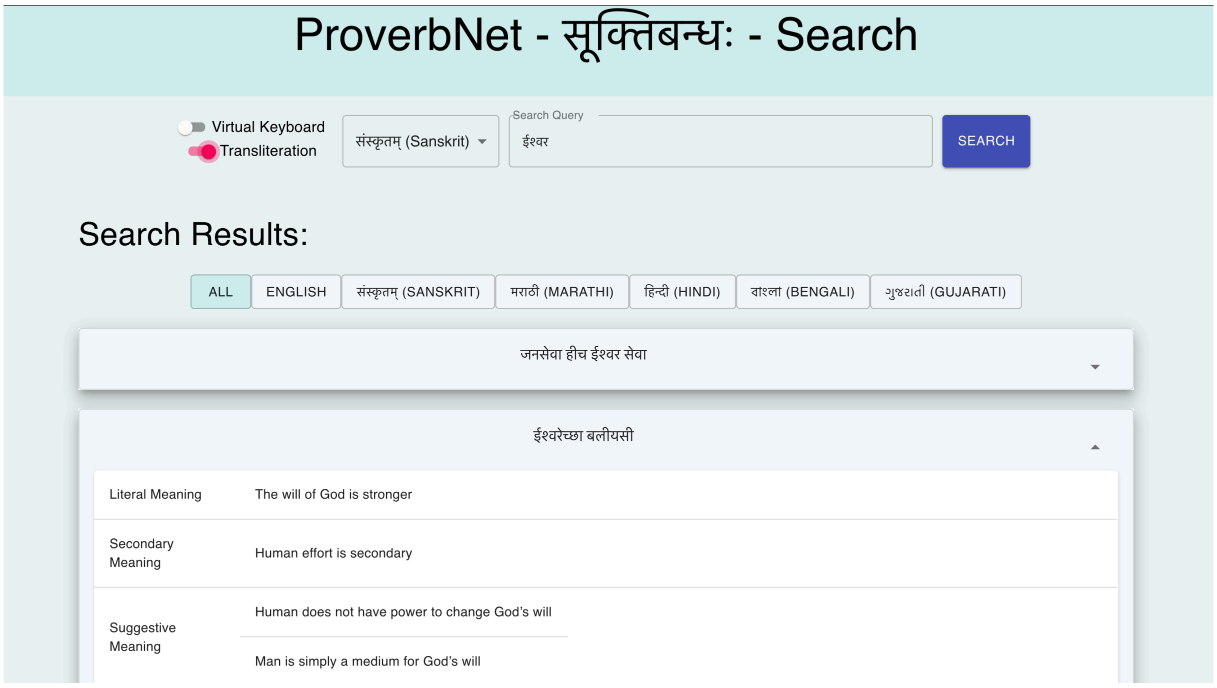
Now, after the data is recorded in the data entry interface, it gets added to the database after performing several validity checks. We have modelled the ProverbNet architecture as a relational database with separate tables for appropriate data elements. Every data entry has a distinct reference ID, is the primary key of the information table, and increments automatically. We put the single-valued entries, namely, literal meaning, secondary meaning, type of proverb, cognition, references, specific observations, historical specifics, social and cultural specifics, lingual specifics, and each of their languages in a single table called the information table. We store the remainder of the data elements, i.e. the ones that have more than one multiplicity, namely proverbs, suggestive meanings, the process of cognition, keywords and categories, classification, and examples in separate tables. This arrangement of data is a natural choice and benefits the search algorithm, which we will explain in the next section.

5 Search

We have designed a search interface (figure 3) to search for proverbs in the desired language. The search takes input a single or a multi-word query in a language specified by the user. The search interface has a transliteration option to type in English, and the text automatically gets transliterated to the selected language. It also has a virtual keyboard to type in the desired language. On submitting, the interface sends the search request to the back-end server for processing. On getting the response, we display the search results in a convenient format inspired by the Indo Wordnet (Bhattacharyya, 2010). The search interface has buttons to filter output proverbs by language. The proverbs are displayed as cards and have a button to expand them and display their stored metadata. It also has a feature to perform an in-depth search and get more results. Users can also specify the type of the query to search specific metadata.

User searches can vary from getting a proverb by entering a part of it to find the processes of cognition of proverbs related to a specific theme. One can search for proverbs by typing in the entire proverb (bhavitavyānām dvārāṇi bhavanti sarvatra |). If they do not know the entire proverb, they can type a few words of the proverb as well (bhavitavyānām bhavanti). If they want to find out a proverb with a given meaning, they can type the meaning of the proverb and get the proverbs that have the entered meaning. The meaning can be literal, secondary or suggestive (No one can change the destiny, or destiny is inevitable). Therefore, ProverbNet acts as a thesaurus of proverbs as well. One can search for proverbs containing specific keywords (bhavitavya, dvāra) or their categories (fate). To get complete śloka, or get proverbs written in specific reference, or get entire proverbs by a particular author, one can enter the reference name or the author name (Śhākuntalam, Kālidāsa). Users can get proverbs that have similar processes of cognition, classification, and other metadata.

The output of a sample query is shown in figure 3. The search currently uses a two-step algorithm to get the relevant output. We assume that the user will generally enter keywords, classification, or a part of the proverb as the search query. The first step of the algorithm is to match proverb prefixes (includes full proverb) with the search query, match exact keywords or categories, and classification. Make individual queries on each table and collate the results. If there are matches, return them. If no matches are present, execute the second step of the search. Another way to trigger the second step is to click the button called 'more results' provided in the search interface. The second step is to per-


ProverbNet - सूक्तिबन्धः - Search

Search Query: ईश्वर

Search Results:

जनसेवा हीच ईश्वर सेवा

ईश्वरेच्छा बलीयसी

Literal Meaning	The will of God is stronger
Secondary Meaning	Human effort is secondary
Suggestive Meaning	Human does not have power to change God's will Man is simply a medium for God's will

Figure 3: Search interface

form a broader search. We break the search query into individual words and remove certain words (e.g. a, and, the), search each word in all of the fields of the database, and return those proverbs not returned in the previous step. This step yields a more extensive collection of proverbs, and they are lesser relevant to the search query than those returned in the first step. We also have another selector in the interface that to specify the type of search query if needed (e.g. meaning, classification) to get more relevant and narrower search results. Selecting the option enables search in only those specific fields.

6 Applications

There are quite a few possible applications of ProverbNet. ProverbNet can be used as an online resource for authors and readers to understand proverbs, as the literal translation may not lead to a comprehensive understanding. Furthermore, The current state-of-the-art systems fail to translate proverbs correctly, and there is a need for such a multilingual resource for correct automatic translation of proverbs. Also, ProverbNet will help get closer to solving the problem of word sense disambiguation. ProverbNet can also become a learning resource as an educational application. Finally, ProverbNet will be useful to get equivalent proverbs in different languages in one place.

7 Future Work

We are adding proverbs from other languages to ProverbNet. The design of ProverbNet ensures little changes to add newer languages. We plan to implement deep learning models to identify proverbs and use ProverbNet for automatic translation. We are also working towards clustering the different entries to create newer entries, so the equivalent proverbs in different data entries get clubbed together. For users having difficulty reading the words in the language present in the database, we plan to add automatic transliteration and translation of metadata in different languages.

References

V.B. Bhagwat. 1985. Paniniy vyakran ani bhasha tatvdnyan.

Pushpak Bhattacharyya. 2010. Indowordnet. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

D. T. Bhosale. 2016. Lopalelyā suvarṇamudrā: Haravata cālalelyā arthapūrṇa mhaṇī'. *Sakala Papers Private Li., Puṇe.*

J. Bronkhorst. 2005. Bhaṭṭoji dikshit on sphota. *Journal of Indian Philosophy*, page 3–41.

J. Brough. 1953. Some indian theories of meaning. *Transaction of the Philological Society*, 52(1):161–176.

Marcus Coinnigh. 2015. *Structural Aspects of Proverbs. In H. Hrisztova-Gotthardt, M. Aleksa Varga (Eds.), Introduction to Paremiology: A Comprehensive Guide to Proverb Studies. Berlin: de Gruyter.*, pages 112–132.

Todd Ferretti, Christopher Schwint, and Albert Katz. 2007. Electrophysiological and behavioral measures of the influence of literal and figurative contextual constraint on proverb comprehension. *Brain and language*, 101:38–49.

C. D. Fellbaum D. Gross K. Miller G. A. Miller, R. Beckwith. 1990. Wordnet: An online lexical database. *Internal Journal of Lexicography*, 3(4):235–244.

Chitra Garg and Lalit Goyal. 2014. Automatic extraction of idiom , proverb and its variations from text using statistical approach.

R. W. Gibbs and D. Beitel. 1995. What proverb understanding reveals about how people think. *Psychological Bulletin*, 118(1):133–154.

P. Grzybek. 2014. *4 Semiotic and Semantic Aspects of Proverb, in: H. Hrisztova-Gotthardt, M. Aleksa Varga (eds).*

R.P. Honeck. 2016. *A Proverb in Mind: The Cognitive Science of Proverbial Wit and Wisdom.* Psychology Press.

R.P. Honeck and J.G. Temple. 1994. Proverbs: The extended conceptual base and great chain metaphor theories. *Metaphor and Symbolic Activity*, 9(2):85–112.

S. Jagannath. 2009. Abhanakajagganatha. raghvendra matham mantralayam.

M. Kulkarni, C. Dangarikar, I. Kulkarni, A. Nanda, and P. Bhattacharya. 2010. Introducing sanskrit wordnet. In *The 5th International Conference of the Global WordNet Association (GWC-2010), 31st Jan-4th Feb.*

M. Kuusi. 1972. *Towards an International Type-system of Proverbs.* Number nos. 211-218 in FF communications. Suomalainen tiedeakatemia.

Outi Lauhakangas. 2001. The matti kuusi international type system of proverbs. *Folklore Fellows' Communications, 275. Helsinki: Academia Scientiarum Fennica.*

Outi Lauhakangas. 2013. The matti kuusi international database of proverbs. *Oral Tradition*, 28.

Syahron Lubis. 2019. The universality and uniqueness of proverb and its impact on translation. In *UNNES International Conference on English Language Teaching, Literature, and Translation (ELTLT 2018)*. Atlantis Press.

Rev. Manwaring. 1899. Marathi proverbs translated and collected. *Clarendon Press, Oxford.*

Wolfgang Meider. 2004. *Proverbs. A Handbook.* Westport, Connecticut: Greenwood Press.

W. Mieder, W. Mieder, and A. Dundes. 1994. The Wisdom of Many: Essays on the Proverb (Vol. 1994).

Vishvanath Naravane. 1978. Bhārtiya vichārsādhanā prakāshan. *Bhārtiya Vichārsādhanā Prakāshan, Pune.*

Neal R. Norrick. 2015. 1 subject area, terminology, proverb definitions, proverb features. *Introduction to Paremiology: A Comprehensive Guide to Proverb Studies*, pages 7–27.

Amanda P. Rassi, Jorge Baptista, and Oto Vale. 2014. Automatic Detection of Proverbs and their Variants. In *3rd Symposium on Languages, Applications and*

Technologies, volume 38 of *OpenAccess Series in Informatics (OASIcs)*, pages 235–249, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Brahmaleen K. Sidhu, Arjan Singh, and Vishal Goyal. 2010. Identification of proverbs in hindi text corpus and their translation into punjabi. *Journal of Computer Science and Engineering*, 2.

D.A. Swinney. 1979. The access and processing of idiomatic expressions. *Journal of Verbal Learning and Verbal Behaviour*, page 523–534.

J.G. Temple. 1999. Proverb comprehension: The primacy of literary meaning. *Journal of Psycholinguistic Research*, 28(1):41–70.

Yulia Tsvetkov and Shuly Wintner. 2014. Identification of multiword expressions by combining multiple linguistic information sources. *Computational Linguistics*, 40(2):449–468.

Roger D. Woodard. 2008. *The Ancient Languages of Asia and the Americas.*, pages 1–2. Cambridge University Press.

Appendix A: Process of Cognition Example

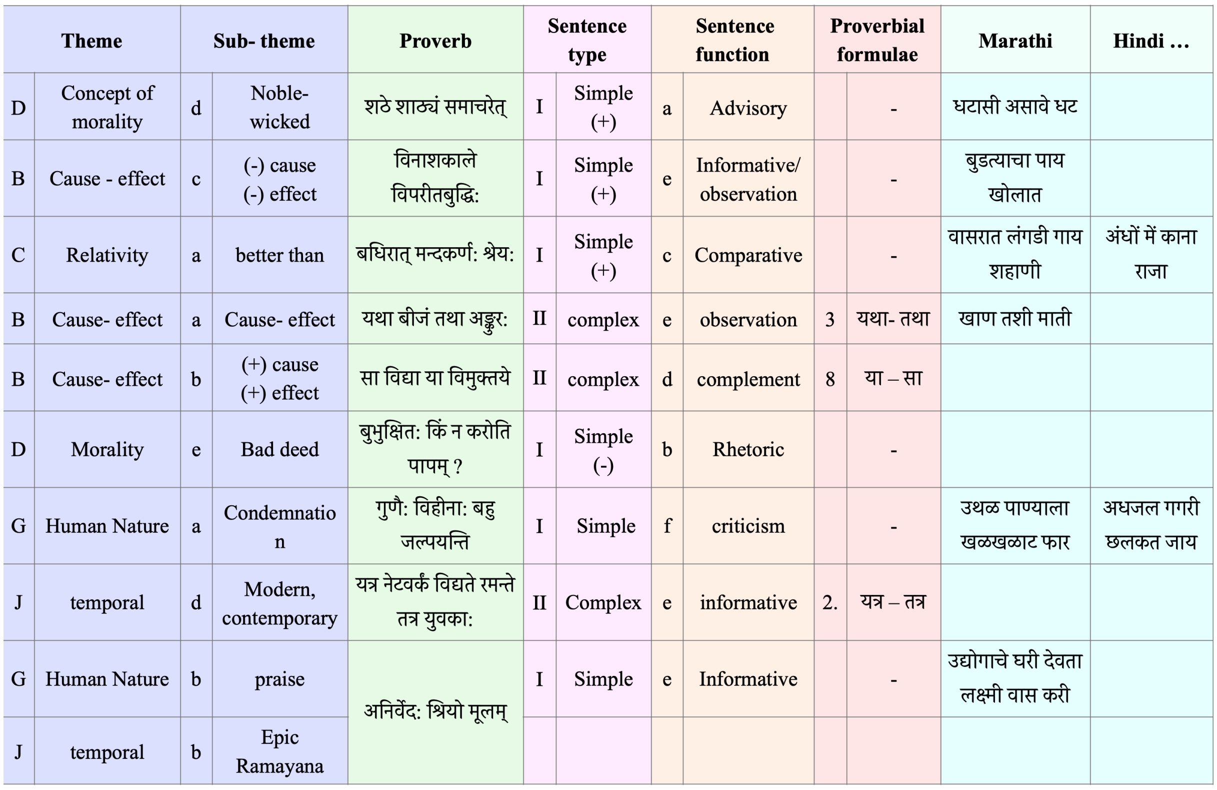
Let us apply process of cognition to: Nirastapādape deshe eraṇḍopi dṛmāyate |

1. Padjnyānm: Nirastapādape deshe eraṇḍopi dṛmāyate

2. padārthjnyānm – After identifying all the words either by their sounds or by the sequence of alphabets, next step is to understand their meanings. The next step is understanding the vṛtti, i.e. the relation of word and word meaning. To understand the meanings of the word we have to rely on the ṣakti or the primary meaning or the denotative power. To understand the ṣakti let us consider the different ways of ṣaktigraha.
vyākaraṇa - Grammar. drumāyate - generally in Sanskrit verbs originate from roots like verb *gaccha* – root *gam*, verb *yaccha* – root *da*. However, there are certain verbs created from Nouns which are called nāmadhātū. They derive a root from a noun which they consider to be the most suitable, the most appropriate to convey their thought. drumāyate: takes the place of a tree, pādapa: it is a compound, pādai: pibati | (upapada tatpuruṣa) one who drinks through their legs – a tree which drinks water through its roots.
koṣa: – Dictionary. *nirasta*– deprived or void of,*eraṇḍa:* castor bean plant. Thus the primary meaning of the proverb attained by the abhidhā ṣakti is: In a place where there is no vegetation, even a castor plant takes the place of a tree. This would have been called a mere description of a castor plant. But since we use the words *eraṇḍopi dṛmāyate*, even a castor plant takes the place of a tree indicates that this sentence wants to say something more than its literal meaning. Thus here abhidhā ṣakti does not suffice. We have to go to the next level, i.e. lakṣaṇā or figurative meaning. But before going to the secondary meaning let us examine whether the words in the given frozen expression satisfy the conditions of ākaṇkṣā, yogyatā, Sannidhi.

Since words are occurring in immediate sequence, condition of Sannidhi (phonetic contiguity) is fulfilled. In this sentence, we have a verb, related kartā to that verb and the other related words also tell us their connection with the verb. So the condition of ākaṇkṣā syntactic expectancy) is also fulfilled. In this expression eraṇd dṛmāyate these two words are incompatible. Because the plant of eraṇd grows up to two to four feet, it is a plant with a small number of leaves and not the tree. Still, it is called a tree. Thus these are incompatible with each other. Though there is no compatibility of words in this expression, this expression is not considered meaningless. It is generally observed that many of the proverbs hold incompatible words and still express deep meaning. The deliberate use of incompatible words in proverbs makes us more attentive towards its meaning. Now let us move to the secondary meaning.

3. The primary meaning attained by the denotative power does not seem meaningful, so we consider the next level, i.e. the secondary meaning or the lakṣaṇā. The secondary meaning of the proverb/ meaning attained by the lakṣaṇā is : When everyone around is of mediocre capacity, one with even a few accolades becomes the hero. The secondary mean-



Theme		Sub- theme		Proverb	Sentence type		Sentence function		Proverbial formulae		Marathi	Hindi …
D	Concept of morality	d	Noble-wicked	शठे शाठ्यं समाचरेत्	I	Simple (+)	a	Advisory		-	घटासी असावे घट	
B	Cause - effect	c	(-) cause (-) effect	विनाशकाले विपरीतबुद्धिः	I	Simple (+)	e	Informative/ observation		-	बुडत्याचा पाय खोलात	
C	Relativity	a	better than	बधिरात् मन्दकर्णः श्रेयः	I	Simple (+)	c	Comparative		-	वासरात लंगडी गाय शहाणी	अंधों में काना राजा
B	Cause- effect	a	Cause- effect	यथा बीजं तथा अङ्कुरः	II	complex	e	observation	3	यथा- तथा	खाण तशी माती	
B	Cause- effect	b	(+) cause (+) effect	सा विद्या या विमुक्तये	II	complex	d	complement	8	या – सा		
D	Morality	e	Bad deed	बुभुक्षितः किं न करोति पापम् ?	I	Simple (-)	b	Rhetoric		-		
G	Human Nature	a	Condemnatio n	गुणैः विहीनाः बहु जल्पयन्ति	I	Simple	f	criticism		-	उथळ पाण्याला खळखळाट फार	अधजल गगरी छलकत जाय
J	temporal	d	Modern, contemporary	यत्र नेटवर्कं विद्यते रमन्ते तत्र युवकाः	II	Complex	e	informative	2.	यत्र – तत्र		
G	Human Nature	b	praise	अनिर्वेदः श्रियो मूलम्	I	Simple	e	Informative		-	उद्योगाचे घरी देवता लक्ष्मी वास करी	
J	temporal	b	Epic Ramayana									

Figure 4: Classification Example

ing tells us the implied meaning of the proverb. The same proverb can be uttered in multiple situations by different speakers. Lakṣaṇā does not cover subjectivism as lakṣaṇā limits itself to figurative meaning. Hence we have to go further and find out the suggestive meaning (vyanjanā) expressed through this proverb. There can be many suggestive meanings depending upon the situation, the intention of the speaker, the way the listener receives it.

4. Suggestive Meanings: It is quite natural to compare oneself to his surroundings and feel a sense of accomplishment. Calling oneself better than those around, aids only in boosting the ego. If the general average of the company is kept low, one may not even know there are bigger things out there in the world. People should raise the bar of self-assessment by keeping the company of noble people.

5. Cognition of the Proverb: Be an achiever in harsh surroundings than being the best amid trivial situations.

Appendix B: Classification of Proverbs

An example of classification is shown in figure 4. Following is the detailed explanation of classification that we use in ProverbNet.

B.1 Structural Classification

1. Sentence Type: Proverbs appear in a variety of different sentence types; from a syntactic perspective, these sentences may be classified into simple, and non-simple (compound, complex) (Coinnigh, 2015). Simple sentences are typically simple, declarative, non-oppositional, and they do not contain any stylistic markers. E.g. ati sarvatra varjayet (affirmative) , amantram akṣram nāsti (negative). compound/complex sentence are sentences containing one or more dependent clauses in addition to the main clause. E.g. yādṛṣm vapate beejam tādṛṣm labhte falam, yah kriyāvān s pandith |

2. Sentence Functions: We classify sentences based on the function they perform in the communication. Proverbs may belong the following categories.

(a) Advisory/potential – in Sanskrit the potential (vidhyartha) form is used to denote advisory function. E.g. ati sarvatra varjayet| kaṇtakenaiva kaṇtakam uddharet |

(b) Rhetoric - A question asked to create a dramatic effect or to make a point rather than to get an answer. E.g. kṛshe kasyasti sauhṛdam? andhasya deepen kim?

(c) Comparative – Denotes comparison between two objects. In Sanskrit the suffixes we use tara, īyasa, varaṃ to denote comparison. E.g. Janani janmabhumischa swargādpi gariyasi | Varam sarpo n durjanah |

(d) Complement (x is y) - It is a word, phrase, or clause that is necessary to complete the meaning of a given expression. In Complement sentence type of proverb, we use one object to denote another object. E.g. yah kriyāvān sah pandith| guṇāh pujāsthanam|

(e) Informative sentence – a sentence which provides general information. E.g. jalbindu nipaten kramaṣah puryate ghath| sāhase shree prativasti|

(f) criticism - proverbs that criticise things, people, actions, behaviour. E.g. kṛśe kasyāsti sauhṛdam|

(g) observation - proverbs that contain general observations. E.g. Fatātopo Bhayankarh |

(h) comment - proverbs that provide general comments, these proverbs can sometimes also be classified as observations. E.g. Maunm sarvārthsadhanm|

(i) suggestion - It is an advice, suggestion. E.g. Mule Kuthār|

B.2 Proverbial Formulae

If applicable, we classify proverbs according to the following formulae:

1. Itah.. that – ito vyāghrh tato tati
2. yatra.. tatra – yatra yatra dhumo tatra tatra vahni
3. yathā.. tathā - yathā rājā tathā prajā
4. yādṛṣm.. tādṛṣm - yādṛṣm vapate beejam tādṛṣm labhte falam
5. x vā.. y vā - naro vā kunjaro vā
6. x vā y.. na vā - dātā bhavti vā na vā
7. na x.. na y - kāmāturaṇām na bhayam na lajjā
8. yah/yā.. sah/sā - yah kriyāvān sah pandith

It is seen that structurally Marathi proverbs are comprised of two distinct parts. Very often a distinct rhyme scheme is seen in these two parts of a proverb. There is no specific theory that explains the reason for such a rhyme scheme based structure. However since the proverbs have their roots in colloquial (oral) use of language, one may conclude that owing to the oral nature of proverbs and its brief, concise and crisp structure rhyme scheme might have been incorporated into proverbs. We classify proverbs as rhyming or not rhyming, and following are some examples of rhyming proverbs.

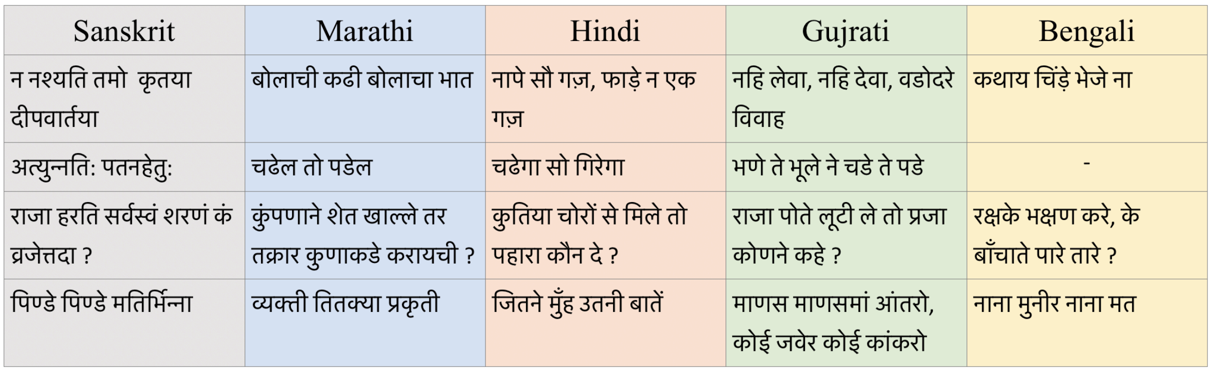
- ati udār to sadā nadār
- sāpalyamadhye wāgh sāpade, bāyakā mule māriti khade
- dev tāri tyālā kon māri
- khāin tar tupāshi nāhitar upāshi
- āiji rāṇi bāyji rāṇi, toṇd dhuvāyalā kon deil pāṇi

B.3 Thematic Classification

(A) Natural Elements: human behaviour explained through natural elements
  (a) water, fire, earth, sea, soil, flora
  (b) animals
  (c) weather

(B) Cause-Effect Relationship
  (a) cause-effect
  (b) positive cause leading towards positive effect
  (c) negative cause leading towards negative effect
  (d) negative cause (contribution) leading towards a positive output

(C) Relativity



Sanskrit	Marathi	Hindi	Gujrati	Bengali
न नश्यति तमो कृतया दीपवार्तया	बोलाची कढी बोलाचा भात	नापे सौ गज़, फाड़े न एक गज़	नहि लेवा, नहि देवा, वडोदरे विवाह	कथाय चिंड़े भेजे ना
अत्युन्नति: पतनहेतु:	चढेल तो पडेल	चढेगा सो गिरेगा	भणे ते भूले ने चडे ते पडे	-
राजा हरति सर्वस्वं शरणं कं व्रजेत्तदा ?	कुंपणाने शेत खाल्ले तर तक्रार कुणाकडे करायची ?	कुतिया चोरों से मिले तो पहारा कौन दे ?	राजा पोते लूटी ले तो प्रजा कोणने कहे ?	रक्षके भक्षण करे, के बाँचाते पारे तारे ?
पिण्डे पिण्डे मतिर्भिन्ना	व्यक्ती तितक्या प्रकृती	जितने मुँह उतनी बातें	माणस माणसमां आंतरो, कोई जवेर कोई कांकरो	नाना मुनीर नाना मत

Figure 5: Equivalent proverbs in different languages

(a) one is better than other
(b) one is bigger than other

(D) Concept of Morality
  (a) good – bad
  (b) pride – humility
  (c) youth – old age
  (d) noble – wicked
  (e) sin – good deed
  (f) virtue – vice

(E) Group Behavioural Observations (subject to morality of the respective community)
  (a) quality having a positive connotation
  (b) quality having a negative connotation

(F) Human Attitude
  (a) optimistic
  (b) pessimistic

(G) Human Nature
  (a) condemnation: laziness, greed, ego, selfishness, cowardice, treachery, misery, foolishness,
  (b) praise: hardworking, bravery, patriotism, politeness, kindness, generosity, cleverness, truthfulness, dexterity, enterprise, intelligence,

(H) Social Position: Based on,
  (a) class: wealth, poverty, money
  (b) caste: hierarchy
  (c) gender: man-woman relationship
  (d) power : strength (powerful/ privileged) – weakness (powerless/unprivileged)

(I) Universal Truth

(J) Temporal - As most of the Sanskrit proverbs have their origin from various verses and various ancient scriptures, this aspect of classification is essential. We can have four subdivisions under this them. They are:
  (a) from Vedic scriptures: Four Vedas, Upaniṣhads, Brāhman, Āraṇyakas
  (b) from epics: Rāmāyaṇa and Mahābhārata
  (c) from classical literature: plays, poems of renowned poets, Panchatantra, vidurniti, Kautilya arthashāstra, triṣatak by Bhṛtṛhari, subhāshitaratna-bhāndāgāram and others
  (d) modern/ contemporary proverbs: ābhāṇakajagannātha

(K) Relationship :
  (a) relatives
  (b) friends
  (c) social relationship
  (d) relationship with animals, plants, nature

Bypassing Optimization Complexity through Transfer Learning & Deep Neural Nets for Speech Intelligibility Improvement

Ritujoy Biswas
Indian Institute of Technology Jammu
ritujoybiswas@gmail.com

Abstract

This extended abstract highlights the research ventures and findings in the domain of speech intelligibility improvement. Till this point, an effort has been to simulate the Lombard effect, which is the deliberate human attempt to make a speech more intelligible when speaking in the presence of interfering background noise. To that end, an attempt has been made to shift the formants away from the noisy regions in spectrum both sub-optimally and optimally. The sub-optimal shifting methods were based upon Kalman filtering and EM approach. The optimal shifting involved the use of optimization to maximize an objective intelligibility index after shifting the formants. A transfer learning framework was also set up to bring down the computational complexity.

1 Motivation of Research

While much of the research focus has been on improvement in quality of speech signals, certain applications call for proper intelligibility of the speech rather than how pleasing it is to the listener. Thus, the prime motivation of the current research is to ensure that information is not lost to noise and is communicated in a robust manner, especially at very low SNR levels.

2 Key Issues; Identified and Addressed

The most common causative factor of loss in speech intelligibility is the presence of background noise. When noise occupies the same regions of the spectrum as speech, the intelligibility falls drastically. One solution is to shift the formants away from the noisy regions in the spectrum. The result would be that the information content in those formants would also be shifted away from the noise and would thus cease to be afflicted by it. The details of formant shifting are given in (Nathwani et al., 2016).

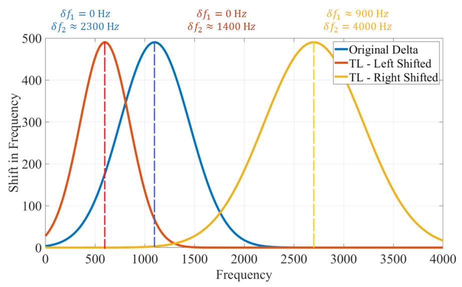


Figure 1: Transformation Function (TF) obtained through CLPSO and Transfer Learning (TL) via left (L) and right (R) shifting.

3 Major Contributions

Following have been the major contributions so far:

1. The first major contribution was the implementation of Comprehensive Learning Particle Swarm Optimization (CLPSO) (Rahmati et al., 2014) for optimization of 5 parameters a Trapezoidal Delta Function based formant shifting framework.

2. One drawback of this optimization was the enormous time-complexity which rendered the approach unsuitable for real-time applications. To address this issue, the next contribution was made. A Transfer Learning (TL) framework was developed which transferred the learning across languages.

3. In parallel, another attempt was made to reduce the time complexity by replacing the Trapezoidal formant shifting with a Gaussian one. Therefore, the next contribution was training a Gaussian delta function using CLPSO to optimize a set of 3 (instead of 5) shaping parameters.

4. Since Gaussian is a statistical shape, it allowed the incorporation of noise in the TL

Table 1: Universal TL performance measured on STOI for EN(TR)→FR(BB) and compared with Trap based TL

SNR	$\mathrm{STOI_O}^{\mathrm{NM}}$	CLPSO (Trap)	TL-Lang (Trap)	CLPSO (Gauss)	TL-Noise (Gauss)	TL-Univ (Gauss)
	FR(*BB*)	$\mathrm{STOI_M}^{FR(BB)\rightarrow FR(BB)}$	$\mathrm{STOI_M}^{EN(BB)\rightarrow FR(BB)}$	$\mathrm{STOI_M}^{FR(BB)\rightarrow FR(BB)}$	$\mathrm{STOI_M}^{FR(TR)\rightarrow FR(BB)}$	$\mathrm{STOI_M}^{EN(TR)\rightarrow FR(BB)}$
-8	0.44	0.57 (+29.54%)	0.47 (+6.82%)	0.58 (+31.82%)	0.57 (+29.54%)	0.56 (+27.27%)
-14	0.31	0.45 (+45.16%)	0.34 (+9.68%)	0.47 (+51.61%)	0.47 (+51.61%)	0.44 (+41.94%)
-26	0.25	0.34 (+36.00%)	0.27 (+8.00%)	0.36 (+44.00%)	0.35 (40.00%)	0.32 (+28.00%)

framework which was earlier not considered. Thus, the next contribution was the development of another TL framework for transferring the learning from one noise environment (*source*) to another (*target*).

5. The final contribution was the amalgamation of both the TL approaches to form a universal TL framework.

4 Methodologies

The foundation of the current work lies in shifting the formants away from noisy regions in speech. To that end, the mathematical representation of this shifting through a Gaussian delta function can be represented as shown:

$$\hat{F} = \begin{cases} \frac{h}{(\mu-\delta f_1)} + F, & if \quad \delta f_1 \leq F < \mu \\ \frac{-h}{(\delta f_2-\mu)} + F, & if \quad \mu \leq F \leq \delta f_2 \\ F, & otherwise \end{cases}$$
$$\delta f_1 = \max(0\,Hz,\ f(<\mu\ @\ TF = 0))$$
$$\delta f_2 = \min(f(>\mu\ @\ TF = 0),\ 4000\,Hz) \tag{1}$$

For the universal TL framework, the formant shifting is applied after the parameters have been modified for the new combination of language and noise at a certain SNR. This universal TL can be mathematically represented as:

$$\mu_T = \mu_S \pm \min(|\frac{F_{T_{avg}}}{F_{S_{avg}}} \times \mu_S|,$$
$$|\frac{\mu_{tn} \sim \mu_{sn}}{\mu_{sn}}\% \text{ of } \mu_S|) \tag{2}$$
$$\sigma_T = \sigma_S \pm \frac{\sigma_{tn} \sim \sigma_{sn}}{\sigma_{sn}}\% \text{ of } \sigma_S$$

The subscripts 'S' & 'T' denote the *source* and *target* transformation functions respectively, and the subscripts 'sn' & 'tn' denote the Gaussian approximation of the magnitude spectra of the two noises being compared. $F_{T_{avg}}$ and $F_{S_{avg}}$ are the average formant frequencies of the *target* and *source* languages respectively. The modification of mean is contributed to by both language and noise transfer. The modification in standard deviation is controlled by noise transfer alone.

5 Experiments and Results

The pipeline of the experiments conducted started with the generation of a Trapezoidal Delta function through CLPSO for a certain combination of Language, Noise type and SNR level. This led to a certain improvement in Short Time Objective Intelligibility (STOI) (Taal et al., 2010) for that specific combination. Thereafter transfer was done across languages and the results were compared with that obtained through direct training through CLPSO. Next, the CLPSO was used to obtain the Gaussian Delta function for a combination as discussed before. Thereafter, transfer was done across noises, followed by a combination of language and noise. These results can be exemplified by one of the cases of training and transfer, as shown in Table 1. The table compares the CLPSO training of both Trapezoidal as well as Gaussian Delta functions, followed by the transfer across Languages and Noises respectively. The final column of the Table shows the results of the universal TL framework working on Languages and Noises simultaneously. It can be seen that the CLPSO training performs better using Gaussian than Trapezoid. Also transfer across Noises improves intelligibility more than transfer across Languages for this particular setup. Furthermore, the universal TL results seem to approach the results as obtained through direct training.

6 Future Plans and Road-map for Thesis

The work done up to this point has provided certain insightful results. These results will be used as prior information to train a Neural Network to develop a fully autonomous system that is scalable in terms of applications. Thus, the thesis is projected to be consisting of two parts. The former dealing with intelligibility improvement of speech using basic machine learning and optimization, while the latter handling these issues through the employment of deep neural networks.

References

Karan Nathwani, Morgane Daniel, Gaël Richard, Bertrand David, and Vincent Roussarie. 2016. Formant shifting for speech intelligibility improvement in car noise environment. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5375–5379. IEEE.

Meysam Rahmati, Reza Effatnejad, and Amin Safari. 2014. Comprehensive learning particle swarm optimization (clpso) for multi-objective optimal power flow. *Indian Journal of Science and Technology*, 7(3):262–270.

Cees H Taal, Richard C Hendriks, Richard Heusdens, and Jesper Jensen. 2010. A short-time objective intelligibility measure for time-frequency weighted noisy speech. In *International Conference on Acoustics, Speech and Signal Processing*, pages 4214–4217. IEEE.

Design and Development of Spoken Dialogue System in Indic Languages

Shrikant Malviya
Department of Information Technology
Indian Institute of Information Technology Allahabad
Prayagraj, India
s.kant.malviya@gmail.com

Abstract

Based on the modular architecture of a task-oriented Spoken Dialogue System (SDS), the presented work focussed on constructing all the system components as statistical models with parameters learned directly from the data by resolving various language-specific and language-independent challenges. In order to understand the research questions that underlie the SLU and DST module in the perspective of Indic languages (Hindi), we collect a dialogue corpus: Hindi Dialogue Restaurant Search (HDRS) corpus and compare various state-of-the-art SLU and DST models on it. For the dialogue manager (DM), we investigate the deep-learning reinforcement learning (RL) methods, e.g. actor-critic algorithms with experience replay. Next, for the dialogue generation, we incorporated Recurrent Neural Network Language Generation (RNNLG) framework based models. For speech synthesisers as a last component in the dialogue pipeline, we not only train several TTS systems but also propose a quality assessment framework to evaluate them.

1 Introduction

Recently, substantial improvements in speech recognition performance have enticed the research community to build natural conversational interfaces in the form of a spoken dialogue system (SDS) (Jurafsky and Martin, 2019). This paper is concerned broadly with designing a complete spoken dialogue system in an Indic language scenario, i.e. Hindi. No significant work has been done earlier to promote the research and development of a Hindi spoken dialogue system. Hence, it becomes critical for the current work to address the issues and challenges unveiled for the Hindi language through introducing new datasets, methods and measures to build and evaluate all the integral modules of the Hindi SDS.

A typical SDS structure is based on a modular pipeline design connecting five principal components in a specific order (Pieraccini and Huerta, 2005): Automatic Speech Recognition (ASR), Spoken Language Understanding (SLU), Dialogue Manager (DM), Natural Language Dialogue Generation (NLDG) and Text-To-Speech Synthesiser (TTS). The work presented in this paper demonstrates how these components are developed individually and integrated at the end to develop a real-world spoken dialogue system in Hindi.

In a statistical spoken dialogue system, the aim is to replace each of the aforementioned components with a statistical model with parameters estimated from data (Young, 2002, 2010). The overall goal is to build a data-driven dialogue system with the ability to get improved over time and be perceived as behaving human-like by the users. The components of such systems are based on statistical methods, i.e. probabilistic distribution, neural network models, which allow them to handle uncertainty in both their inputs and outputs (Young et al., 2013; Zhang et al., 2001).

As the Hindi text contains lots of lexical/morphological ambiguities, therefore, it becomes a key challenge for DST and NLDG models to appropriately detect the DAs, understand the utterances and generate natural responses. Hindi is very rich in inflectional morphology. There is usually a limit of 8-9 inflected word forms of nouns in English (Yule, 2020), but in Hindi, it is more than 40 (Goyal and Lehal, 2008; Vikram, 2013). The way a language is spoken and written gets changed from place to place. It leads to the introduction of variations where the meaning of a sentence is the same, but the way to express gets changed (Geeraerts et al., 2012).

Other language-related challenges that a Hindi SDS have to deal with are code-mixing (Ramanathan et al., 2009), hidden information (Miller et al., 1994), echo-words (Mohan, 2006), etc. Code-mixing is the mixing of more languages in the conversation. There are many cases in the corpus where the user had expressed some words from English during the conversation. (Example: “मुझे कम रेंज वाले रेस्तरां की तलाश है।” (“I am looking for low (cost) range restaurants.”)). Here the word “रेंज” (range) is an English word that gives an indication of the cost.

2 Contributions

This research contributes at the following levels:

1. HDRS corpus: It raises the key research questions that underlie the *SLU* and *DST* module in building a Hindi dialogue system for the restaurant domain. Both traditional embeddings, i.e. *Word2Vec*, *GloVE* & *FastText* as well as *BERT* based embeddings are experimented.

2. A2CER: We incorporate the *advantage actor-critic with experience replay* (A2CER) algorithm (Wang et al., 2017) for dialogue policy learning which has recently been shown to be performing well on simple gaming environments and compare its performance with other state-of-the-art methods on a dialogue task.

3. Hindi NLG corpus: A corpus of unstructured input-output pair of *dialogue-act* (system's) and corresponding *natural response* is collected and released. The *RNNLG framework* based models are experimented on it.

4. Quality Assessment of TTS: A novel evaluation framework: *LBOE (Learning-Based Objective Evaluation)*, is developed for the quality assessment of various TTS systems. For the experiment, several “off-the-self” TTS systems: USS, HMM, CLU and DNN, have been trained from scratch.

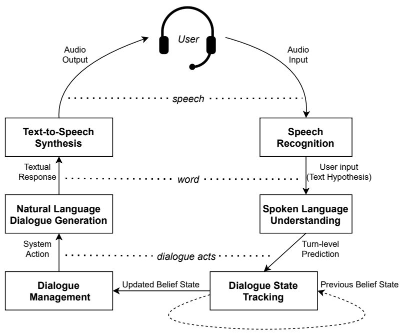


Figure 1: The pipeline of the core components in statistical spoken dialogue systems.

3 SILPA[1]: a Hindi SDS

We design our Hindi SDS by dividing it into five modules in a pipeline architecture (Pieraccini and Huerta, 2005) and connecting them in a specific order, as shown in Figure 1. The remainder of the section discusses the contributions specific to these modules.

3.1 HDRS: Language Understanding & State Tracking

For the empirical analysis of *language-specific* and *language-independent* challenges in dialogue state tracking, we release a dialogue corpus (HDRS) to train SLU/DST models in a *new language Hindi* with better annotations and high language-variability with significant corpus size (Malviya et al., 2021).

An SLU/DST component takes a sentence as input and maps it to an output dialogue act representing underlying semantics. For example, the utterance:
‘मैं एक महंगा रेस्तरां खोज रहा हूँ जहाँ राजस्थानी खाना मिलता हो।’
can be represented as:
`inform(type=restaurant,price range=महंगा,food=राजस्थानी).`

3.2 Modelling Dialogue Management

We model the dialogue policy with RL approaches where the system's goal is to choose a sequence of system responses (actions) given the observed belief state achieving the maximum total reward, whereby the success of the dialogue mainly determines the reward. In this work, we have explored and investigated the current state-of-the-art methods

[1]SILPA (SILPAssistant): The name is based on our Lab's name SILP (Speech, Image & Langauge Processing) Lab

of policy optimisation for a task-oriented dialogue system, i.e. GP-SARSA, DQN, A2C. Inspired by (Wang et al., 2017), we present a new method that combines the strength of experience-replay in A2C (A2CER) policy learning for better dialogue modelling.

3.3 Natural Language Dialogue Generation

Obtaining the dialogue act from the dialogue manager, the Natural Language Dialogue Generation (NLDG) module transforms this abstract semantics notation (system dialogue act) back into a text representation (Singh et al., 2019). For example, the dialogue act:

`request(food)`

can be transformed to:

"आप किस प्रकार का भोजन खाना चाहेंगे? "

In our work, we have explored several state-of-the-art RNNLG-based models with discussing their performances on language-related (Hindi) challenges. All the models are experimented on our own Hindi dataset, collected on the restaurant domain.

3.4 Speech Synthesis & Quality Evaluation

At the last step in the SDS pipeline, the speech synthesis component converts the chosen text or the symbolic linguistic representation into a speech waveform. For the current study, we aim to cover leading TTS technologies as used in research as well as state-of-the-art commercial systems. Both TTS datasets, i.e. IIT-Madras, CMU, are used to build four types of unmodified "off-the-shelf" TTS systems: Unit selection synthesis (USS), Hidden Markov Model (HMM), Clustergen synthesis (CLU) and DNN synthesis (Tacotron 2). This forms the corpus and sets the background for our proposed 'LBOE' framework.

4 Dialogue Agent & Web Interface

We incorporated and adapted the multi-domain statistical dialogue System toolkit *PyDial-Toolkit* (Ultes et al., 2017) to build our dialogue agent "SILPAssistant". The **Agent** is the main component responsible for the dialogue interaction. The general architecture of the dialogue system with a speech interface is shown in Figure 2. The Agent can communicate to the user in both texts as well as speech. For the text-based interaction, **Texthub** utility is provided, which simply connects the Agent to a terminal. To enable speech-based dialogue, the **Dialogue-Server** works as an interface between the Agent and the Speech-Client.

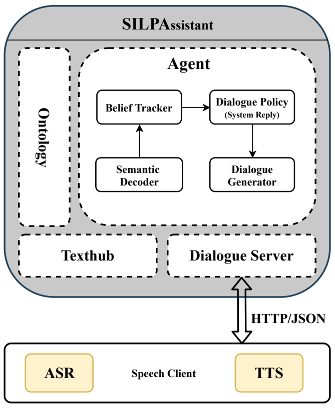


Figure 2: The general architecture of SILPA. The Agent resides at the core, and the interfaces Texthub, Dialogue Sever provide the link to the environment.

5 Conclusion & Future Studies

The current work has examined the challenges of developing a conversational system built upon native *Indian languages* for a real-world task. The original contributions of this thesis include: the development of an *HDRS* corpus on which various state-of-the-art SLU and DST models, i.e. *NBT*, *GLAD*, *GCE*, *GSAT*, *Simple-BERT* and *SUMBT*, are implemented and compared; the RNNLG models, i.e. *H-LSTM*, *SC-LSTM*, *MSC-LSTM* and *ENC-DEC*, have been experimented and used to train corpus-based NLDG module on a self-collected corpus in an Indic language Hindi; construction of dialogue policy with RL based approaches, i.e. GP-SARSA, DQN, A2C (Actor-Critic), A2CER (proposed), on the user-system act pairs generated by a user simulator; proposing a novel framework *LBOE* for quality assessment of a synthesised speech generated from various TTS engines, i.e. USS, HMM, CLU and DNN.

In the current work, we have explored a unimodal natural-language based dialogue scenario. As the human-to-human conversation is multimodal, involving various linguistic forms and non-verbal signals (Firdaus et al., 2021), a multimodal human-to-computer conversation should therefore be more intuitive.

References

Mauajama Firdaus, Nidhi Thakur, and Asif Ekbal. 2021. Aspect-aware response generation for multimodal dialogue system. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 12(2):1–33.

Dirk Geeraerts, Stefan Grondelaers, and Peter Bakema. 2012. *The structure of lexical variation: Meaning, naming, and context*, volume 5. Walter de Gruyter.

Vishal Goyal and Gurpreet Singh Lehal. 2008. Hindi morphological analyzer and generator. In *2008 First International Conference on Emerging Trends in Engineering and Technology*, pages 1156–1159. IEEE.

Dan Jurafsky and James H Martin. 2019. Chatbots and dialogue systems. In *Speech and Language Processing (3rd draft ed.)*. Stanford University.

Shrikant Malviya, Rohit Mishra, Santosh Kumar Barnwal, and Uma Shankar Tiwary. 2021. HDRS: Hindi dialogue restaurant search corpus for dialogue state tracking in task-oriented environment. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:2517–2528.

Scott Miller, Robert Bobrow, Robert Ingria, and Richard Schwartz. 1994. Hidden understanding models of natural language. In *Proceedings of ACL*, pages 25–32.

Shailendra Mohan. 2006. Echo-word formation in hindi. *Indian Linguistics*, 67:119–126.

Roberto Pieraccini and Juan Huerta. 2005. Where do we go from here? research and commercial spoken dialog systems. In *Proceedings of SIGDIAL*, pages 1–10.

Ananthakrishnan Ramanathan, Hansraj Choudhary, Avishek Ghosh, and Pushpak Bhattacharyya. 2009. Case markers and morphology: Addressing the crux of the fluency problem in english-hindi smt. In *Proceedings of ACL*, pages 800–808.

Sumit Singh, Shrikant Malviya, Rohit Mishra, Santosh Kumar Barnwal, and Uma Shanker Tiwary. 2019. Rnn based language generation models for a hindi dialogue system. In *International Conference on Intelligent Human Computer Interaction*, pages 124–137. Springer.

Stefan Ultes, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Dongho Kim, Iñigo Casanueva, Paweł Budzianowski, Nikola Mrkšić, Tsung-Hsien Wen, Milica Gašić, and Steve Young. 2017. PyDial: A multi-domain statistical dialogue system toolkit. In *Proceedings of ACL*, pages 73–78. Association for Computational Linguistics.

Shweta Vikram. 2013. Morphology: Indian languages and european languages. *International Journal of Scientific and Research Publications*, 3(6):1–5.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. 2017. Sample efficient actor-critic with experience replay. In *Proceedings of ICLR*.

Steve Young. 2002. Talking to machines (statistically speaking). In *Seventh International Conference on Spoken Language Processing*.

Steve Young. 2010. Still talking to machines (cognitively speaking). In *INTERSPEECH*.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. POMDP-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

George Yule. 2020. *The study of language*. Cambridge University Press.

Bo Zhang, Qingsheng Cai, Jianfeng Mao, and Baining Guo. 2001. Planning and acting under uncertainty: A new model for spoken dialogue systems. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, page 572–579.

FinRead: A Transfer Learning Based Tool to Assess Readability of Definitions of Financial Terms

Sohom Ghosh[†], **Shovon Sengupta**[†], **Sudip Kumar Naskar**[⋆], **Sunny Kumar Singh**[‡]
[†]Fidelity Investments, Bengaluru, India
[⋆]Jadavpur University, Kolkata, India
[‡]BITS, Pilani, Hyderabad, India
{sohom1ghosh, ssg.plabon, sudip.naskar, sunnysingh.econ}@gmail.com

Abstract

Simplified definitions of complex terms help learners to understand any content better. Comprehending readability is critical for the simplification of these contents. In most cases, the standard formula based readability measures do not hold good for measuring the complexity of definitions of financial terms. Furthermore, some of them works only for corpora of longer length which have at least 30 sentences. In this paper, we present a tool for evaluating readability of definitions of financial terms. It consists of a Light GBM based classification layer over sentence embeddings (Reimers et al., 2019) of FinBERT (Araci, 2019). It is trained on glossaries of several financial textbooks and definitions of various financial terms which are available on the web. The extensive evaluation shows that it outperforms the standard benchmarks by achieving a AU-ROC score of 0.993 on the validation set.

1 Introduction

The notion of readability assumes a central position in the emerging financial literature on textual analysis. Readability as a concept is difficult to define precisely. Readability can be broadly defined as a measure of how easy a text document is to read. In this exercise we aim to examine the readability measure for terms present in a financial glossary and compare various formula based approaches for measuring readability. These include "Automated Readability Index (ARI)" (Smith and Senter, 1967), "Flesch Reading Index (FRI)" (Flesch, 1948), "Dale-Chall formula (DCF)" (Chall and Dale, 1995) and "SMOG Index Score (SIS)" (Mc Laughlin, 1969). At the very outset, all these measures calculate a readability score based on U.S education system's grade level or years of education a reader might require to understand a text content. We further explore the limitations of these measures in the context of financial terms and develop a transfer learning-based system to measure readability of their definitions.

Inspired by the approach followed by (Chakraborty et al., 2021), we collect financial terms and their definitions from seven different sources. We crawl the data dictionary of a popular financial website, Investopedia[1]. The other six sources are text books related to finance. We extract financial terms and their definitions from glossaries of these books by transforming them to HTML format. The data distribution is: NCERT-"Introductory Macro-economics" (149 records), Investopedia (6204 records), (Samuelson and Nordhouse, 2009) (350 records), (Brealey et al., 2012) (177 records), (Hull, 2003) (531 records), (Bodie and Kane, 2020) (525 records), (Mishkin and Eakins, 2006) (465 records).

Among these sources, we assign a readability score of 1 to the definitions present in NCERT, Investopedia and (Samuelson and Nordhouse, 2009). For the remaining sources we assign a readability score of 0. We do this because the content of the former three sources are simple. They are read by school going kids and people in general. The latter four sources constitute of complex graduate level textbooks. We use 80% data for training and the remaining for validation. Finally, we extract ARI, FRI, DCF and SIS scores for each of the definitions using the textstat[2] library.

2 Model Development & Results

In this section, we discuss various approaches we explored and their performances. Firstly, using standard methods we calculate Area under Receiver Operating Characteristic curve (AU-ROC) which

[1]https://www.investopedia.com/financial-term-dictionary-4769738 accessed on 1st Oct 2021

[2]https://pypi.org/project/textstat/ accessed on 1st Oct 2021

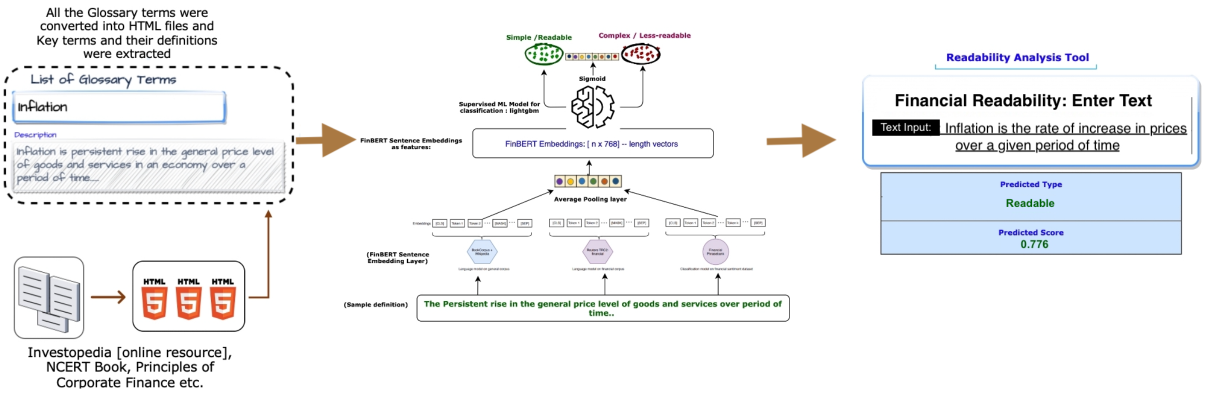

All the Glossary terms were converted into HTML files and Key terms and their definitions were extracted
List of Glossary Terms
Inflation
Description
Inflation is persistent rise in the general price level of goods and services in an economy over a period of time.
HTML
Investopedia [online resource], NCERT Book, Principles of Corporate Finance etc.
Simple /Readable
Complex / Less-readable
Sigmoid
Supervised ML Model for classification : lightgbm
FinBERT Sentence Embeddings as features:
FinBERT Embeddings: [n x 768] -- length vectors
Average Pooling layer
Embeddings
[CLS]
Token 1
Token 2
[MASK]
[SEP]
Token k
(FinBERT Sentence Embedding Layer)
BookCorpus + Wikipedia
Reuters TRC2-financial
Financial Phrasebank
Language model on general corpus
Language model on financial corpus
Classification model on financial sentiment dataset
(Sample definition)
The Persistent rise in the general price level of goods and services over period of time..
Readability Analysis Tool
Financial Readability: Enter Text
Text Input:
Inflation is the rate of increase in prices over a given period of time
Predicted Type
Readable
Predicted Score
0.776

Figure 1: Financial Readability Flow Chart and Tool

is 0.742 for ARI, 0.435 for FRI, 0.413 for DCF and 0.730 for SIS. After that, we represent the definitions numerically using TF-IDF (ngrams: 1 to 4) and train several machine learning based models for classification like Logistic Regression, Random Forest and so on. We also experiment by replacing TF-IDF with "sentence-transformers" based embeddings (Reimers et al., 2019) with FinBERT (Araci, 2019) (768 dimensions). We also try other classification based approaches like XGBoost, CatBoost and lightGBM. We have summarised the model performances on the validation set in Table 1. We perform these experiments on Google Colab. Analysing the results we conclude that a lightGBM (20 min-child samples, 31 num-leaves) based classifier trained over sentence transformers embeddings (Reimers et al., 2019) having FinBERT (Araci, 2019) gives the best performance (AU-ROC 0.993). Moreover, it outperforms all the standard methods of measuring readability (like ARI, FRI, DCF and SIS) in terms of AU-ROC on the validation set.

Our contributions: a) Preparation of a corpus comprising glossaries of financial terms and their definitions b) *FinRead*- a tool to assess the readability of such definitions as shown in Figure 1.

In the future, we want to improve the overall quality of the system by increasing the size and quality of the corpora.

Model	AU-ROC	Model	AU-ROC
T+LR	0.940	F+RF	0.983
T+RF	0.930	F+XG	0.988
F+LR	0.992	**F+LG**	**0.993**

Table 1: Model performance on validation set. T: TF-IDF vectors, F: FinBERT embeddings, LR: Logistic Regression, RF: Random Forest, XG: XGBoost, LG: lightGBM

References

Dogu Araci. 2019. Finbert: Financial sentiment analysis with pre-trained language models.

Zvi Bodie and Alex Kane. 2020. Investments.

Richard A Brealey, Stewart C Myers, Franklin Allen, and Pitabas Mohanty. 2012. *Principles of corporate finance*. Tata McGraw-Hill Education.

Susmoy Chakraborty, Mir Tafseer Nayeem, and Wasi Uddin Ahmad. 2021. Simple or complex? learning to predict readability of bengali texts. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12621–12629.

Jeanne Sternlicht Chall and Edgar Dale. 1995. *Readability revisited: The new Dale-Chall readability formula*. Brookline Books.

Rudolph Flesch. 1948. A new readability yardstick. *Journal of applied psychology*, 32(3):221.

John C Hull. 2003. *Options futures and other derivatives*. Pearson Education India.

G Harry Mc Laughlin. 1969. Smog grading-a new readability formula. *Journal of reading*, 12(8):639–646.

Frederic S Mishkin and Stanley G Eakins. 2006. *Financial markets and institutions*. Pearson Education India.

Nils Reimers, Iryna Gurevych, Nils Reimers, Iryna Gurevych, Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Paul Samuelson and V Nordhouse. 2009. Economics: a textbook.

E A Smith and R. Senter. 1967. Automated readability index. *AMRL-TR. Aerospace Medical Research Laboratories*, pages 1–14.

Demo of the Linguistic Field Data Management and Analysis System - LiFE

Siddharth Singh, Ritesh Kumar, Shyam Ratan, Sonal Sinha
Department of Linguistics, Dr. Bhimrao Ambedkar University, Agra
sidd435@gmail.com ritesh78_llh@jnu.ac.in
shyamratan2907@gmail.com sonalsinha2612@gmail.com

Abstract

In the proposed demo, we will present a new software - Linguistic Field Data Management and Analysis System - LiFE - an open-source, web-based linguistic data management and analysis application that allows for systematic storage, management, sharing and usage of linguistic data collected from the field. The application allows users to store lexical items, sentences, paragraphs, audio-visual content including photographs, video clips, speech recordings, etc, along with rich glossing / annotation; generate interactive and print dictionaries; and also train and use natural language processing tools and models for various purposes using this data. Since its a web-based application, it also allows for seamless collaboration among multiple persons and sharing the data, models, etc with each other.

The system uses the Python-based Flask framework and MongoDB (as database) in the backend and HTML, CSS and Javascript at the frontend. The interface allows creation of multiple projects that could be shared with the other users. At the backend, the application stores the data in RDF format so as to allow its release as Linked Data over the web using semantic web technologies - as of now it makes use of the OntoLex-Lemon for storing the lexical data and Ligt for storing the interlinear glossed text and then internally linking it to the other linked lexicons and databases such as DBpedia and WordNet. Furthermore it provides support for training the NLP systems using scikit-learn and HuggingFace Transformers libraries as well as make use of any model trained using these libraries - while the user interface itself provides limited options for tuning the system, an externally-trained model could be easily incorporated within the application; similarly the dataset itself could be easily exported into a standard machine-readable format like JSON or CSV that could be consumed by other programs and pipelines. The system is built as an online platform; however since we are making the source code available, it could be installed by users on their internal / personal servers as well.

1 Introduction

Linguistic data management and analysis tools have always been a requirement of field linguists. A huge amount of data is collected and analysed by field linguists for a large number of languages including relatively lesser-known, minoritised and endangered languages of the world and these need to be properly stored, analysed and made accessible to the larger community. On the other hand, there are a huge number of languages across the globe (including the kinds mentioned above), whose data is not available for building any kind of language technology tools and applications. In order to tackle this multi-faceted problem of storing, processing, retrieving and analysing the primary linguistic data, an integrated system with an easily-accessible and user-friendly interface aimed at linguists needs to be made available. "`LiFE`" is developed with the intent of providing a practical intervention in the field by making available an organised framework for management, analysis, sharing (as linked data) and processing of primary linguistic field data including development of digital and print lexicons, sketch grammars and fundamental language processing tools such as part-of-speech tagger and morphological analysers. The software provides an easy-to-use, intuitive interface for performing all the tasks and there is an emphasis on automating the tasks as far as possible. For example, given some initial input, the system incrementally trains automated methods for inter-linear glossing of the dataset (which improves as more data is stored in the system) and subsequent generation of sketch grammar as well as NLP tools for the language. Similarly, the system automatically infers and links the entries in the lexicon and interlinear glossed data using Lemon (more specifically OntoLex-Lemon) (McCrae et al., 2017) and Ligt (Chiarcos and Ionov, 2019).

2 Motivation and Features

Linguistic field data storage, management, sharing and linked data generation has largely developed independent of each other. As such while there are quite a few tools and applications aimed at field linguists (or community members interested in fieldwork for their own language) for collection and management of data as well as generating lexicon, such as FieldWorks Language Explorer (FLEx)[1] (Butler and Volkinburg, 2007) (Manson, 2020); Toolbox[2] (Robinson et al., 2007); Lexique-Pro[3] (Guérin and Lacrampe, 2007); WeSay[4] (Perlin, 2012) (Albright and Hatton, 2008) and a few other platforms for archiving and providing access to the data, the prominent ones being Endangered Languages Archive (ELAR)[5] (Nathan, 2010); The Language Archive (TLA)[6] (Cho, 2012); SIL Language and Culture Archive[7], etc. The Open Language Archives Community (OLAC)[8], which is a consortium of over 60 participating linguistic archives of various kinds (including the ones mentioned above and others for storage and access of linguistic data, especially of endangered languages) has also recently joined the Linguistic Linked Data Open Cloud which paves the way for providing a large amount of such data as linked data (Simons and Bird, 2003). However none of the tools and platforms directly provide an interface for storing or (largely) automatically generating the primary linguistic data as linked data or provide a seamless two-way between the NLP tools and libraries and linguistic data management softwares.

On the other hand, the linked data community has developed tools for supporting generation of linked data, especially linked data lexicons. One of the best-known tools for this is **VocBench (VB)**, which is a fully-fledged open-source web-based thesaurus management platform with the feature of collaborative development of multilingual datasets compatible with semantic Web standards. It provides the facilities of generating lexicons, thesauri, and linked data ontologies to the large organisations, companies, and user communities (Stellato et al., 2020). However tools like these focus on generating Linked Data which is generally not very user-friendly for field linguists nor do they provide options for automating the tasks or linking to the NLP ecosystem.

The primary motivation for building this platform is to provide a tool that acts as a bridge between field linguists (who are primarily engaged in data collection from low-resource and endangered languages, building lexicons, writing grammatical descriptions and also producing educational and other kinds of materials for the communities that they work with), linked data community (who are primarily engaged in meaningfully connecting data from different languages and resources using the semantic web techniques) and the NLP community (who primarily makes use of the linguistic data from multiple languages; could potentially provide support in automating the tasks carried out by field linguists; and also provide tools and technologies for the marginalised and under-privileged linguistic communities). As such in its current state the app provides the following functionalities -

- It provides a user-friendly interface for storing, sharing and making publicly available the linguistic field data including interlinear glossed text, lexicon and associated multimedia content.

- It provides reasonable automation for tasks such as generating lexicon, sketch grammar, etc by providing interfaces for training as well as using pre-trained NLP models needed for automating various tasks. The tool currently supports training various algorithms of the scikit-learn and HuggingFace Transformers library as well as using the models trained using these libraries.

- It provides interface for exporting the data in structured formats such as RDF, JSON and CSV that could be directly used for NLP experiments and modelling.

During the demo we will present these features and the interface of the tool in detail and also briefly train the participants in using it.

[1]https://software.sil.org/fieldworks/
[2]https://software.sil.org/shoebox/, https://software.sil.org/toolbox/
[3]https://software.sil.org/lexiquepro/
[4]https://software.sil.org/wesay/
[5]https://www.elararchive.org/
[6]https://archive.mpi.nl/tla/
[7]https://www.sil.org/resources/language-culture-archives
[8]http://www.language-archives.org/archives

3 Presenters

The demo will be given by the developers of this application which include the following -

1. **Ritesh Kumar** is Assistant Professor of Linguistics and coordinator of the masters program in computational linguistics at Dr. Bhimrao Ambedkar University, Agra. he is working in the field of computational linguistics and language documentation and description for over last 10 years. He has conceptualised, mentored and co-developed this app.

2. **Siddharth Singh** is a software engineer and is currently pursuing his MSc in Computational Linguistics from Dr. Bhimrao Ambedkar University. He is the principal developer of the app,

3. **Shyam Ratan** is pursuing his Mphil in Computational Linguistics and is a co-developer of the app.

4. **Sonal Sinha** is pursuing her Mphil in Computational Linguistics and is a co-developer of the app.

References

Eric Albright and John Hatton. 2008. Wesay, a tool for collaborating on dictionaries with non-linguists. *Documenting and revitalizing Austronesian languages*, 6:189 – 201.

Lynnika Butler and Heather Volkinburg. 2007. Review of fieldworks language explorer (flex). *Language Documentation Conservation*, 1.

Christian Chiarcos and Maxim Ionov. 2019. Ligt: An llod-native vocabulary for representing interlinear glossed text as RDF. In *2nd Conference on Language, Data and Knowledge, LDK 2019, May 20-23, 2019, Leipzig, Germany*, volume 70 of *OASICS*, pages 3:1–3:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Julia Cho. 2012. *The Language Archive*. Dramatists Play Service.

Valérie Guérin and Sébastien Lacrampe. 2007. Lexique pro. *Language Documentation Conservation*, 1(2):293 – 300.

Ken Manson. 2020. Fieldworks linguistic explorer (flex) training 2020 (ver 1.1 august 2020).

John P. McCrae, Julia Bosque-Gil, Jorge Gracia, Paul Buitelaar, and Philipp Cimiano. 2017. The ontolex-lemon model: Development and applications. Brno. Lexical Computing CZ s.r.o.

David Nathan. 2010. Archives 2.0 for endangered languages: From disk space to myspace. *International Journal of Humanities and Arts Computing*, 4:111–124.

Ross Perlin. 2012. Wesay, a tool for collaborating on dictionaries with non-linguists. *Language Documentation & Conservation*, 6:181 – 186.

Stuart Robinson, Greg Aumann, and Steven Bird. 2007. Managing fieldwork data with toolbox and the natural language toolkit. *Language Documentation Conservation*, 1.

Gary Simons and Steven Bird. 2003. The open language archives community: An infrastructure for distributed archiving of language resources. *Computing Research Repository - CORR*, 18:117–128.

Armando Stellato, Manuel Fiorelli, Andrea Turbati, Tiziano Lorenzetti, Willem van Gemert, Denis Dechandon, Christine Laaboudi-Spoiden, Anikó Gerencsér, Anne Waniart, Eugeniu Costetchi, and Johannes Keizer. 2020. Vocbench 3: A collaborative semantic web editor for ontologies, thesauri and lexicons. *Semantic Web*, 11:1–27.

Text Based Smart Answering System in Agriculture using RNN

Raji Sukumar A[1], **Hemalatha N**[2], **Sarin S**[1], and **Rose Mary C A**[2]

[1]Techtern Pvt Ltd Kannur. Kerala,
[2]Aloysius Institute of Management and IT (AIMIT),Mangalore University
rajivinod.a@gmail.com, hemalatha@staloysius.ac.in,
sarinsukumar@gmail.com, rosemaryca@gmail.com

Abstract

Agriculture is an important aspect of India's economy, and the country currently has one of the highest rates of farm producers in the world. Farmers need hand holding with support of technology. A chatbot is a tool or assistant that you may communicate with via instant messages. The goal of this project is to create a Chatbot that uses Natural Language Processing with a Deep Learning model. In this project we have tried implementing Multi-Layer Perceptron model and Recurrent Neural Network models on the dataset. The accuracy given by RNN was 97.83%.

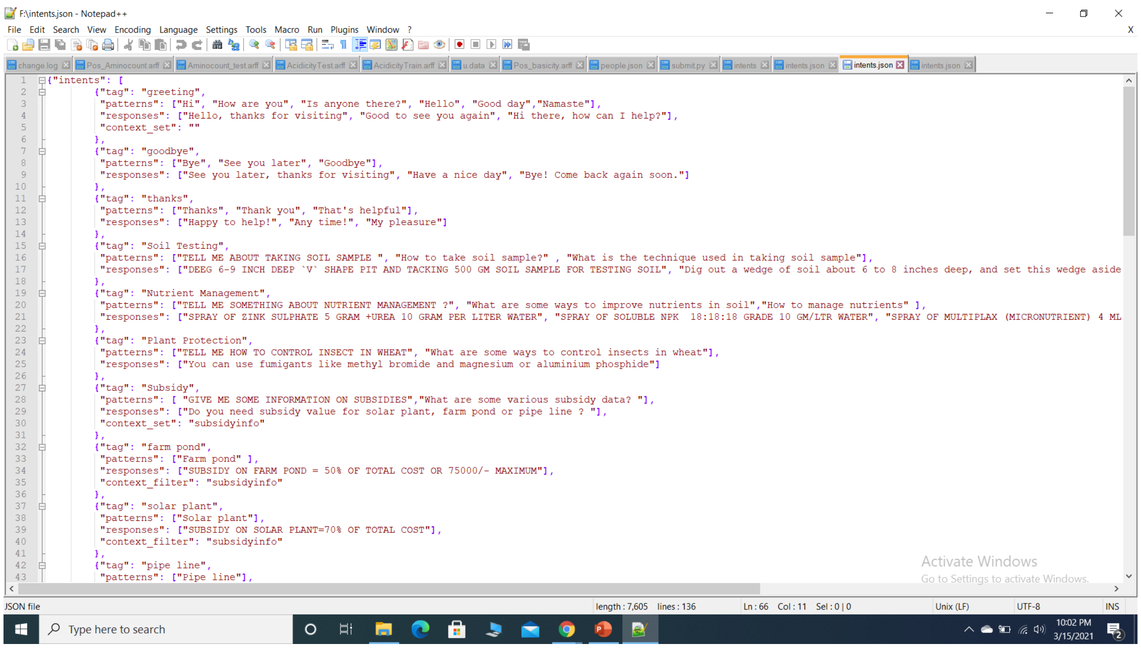
1 Introduction

Agriculture contributes around 16 percent of India's GDP and employs about 52 percent of the country's population, making it a significant part of the country's economic growth. According to the Farmers' Portal, agriculture's rapid expansion is necessary not just for self-sufficiency but also for earning vital foreign exchange. One of the reasons for this is that individuals in the farming industry are relatively sluggish to accept emerging innovations. Field officers have traditionally visited farmlands to give training, guidance, and assistance to farmers. The data demonstrates that mobile connection is increasing at an exponential rate, which helps IT services promote agricultural information. The government is having difficulty disseminating important agricultural information. Furthermore, the difficulties are exacerbated by the dissemination of disinformation. These issues exist as a result of the huge linguistic variety and the rural population's lack of trust in contemporary technologies. In such a situation, using mobile devices to disseminate agricultural information looks to be a viable option [1] (K., 2020). Chat-Bot systems are a type of natural language processing that demands the system to be taught in human language in order to meet the user's demands. Agriculture is the most important sector for a country's development. Farmers are now unaware of the most modern technology and methods employed in agriculture. The challenge of extracting meaningful answers using machine learning techniques has been researched by numerous machine learning specialists, and sophisticated machine learning approaches have been created. These methods are used to obtain the correct answer. We may name this an Agriculture Question Answering System, since the farmer can ask the system a question, and the system will answer (Heller et al., 2005; Beaudry et al., 2019; Sutoyo et al., 2019). With the advancement of technology, farmers must study and address the challenges. As a result, the goal is to create a chatbot system that delivers accurate responses to queries. According to a major study in the field of chatbot systems, there is no agriculture-specific system that can provide precise and rapid answers to farmers' questions. To solve this issue, the suggested system uses the RNN (Recurrent Neural Network) deep learning method to offer accurate responses to the queries asked.

[1]`https://en.wikipedia.org/wiki/Agriculture_in_India/`

2 Problem Statement

The traceability software from Source Trace gives you complete visibility into the agricultural value chain. It has an influence on farmers' lifestyles, helps an organization adopt data-driven agriculture, and fosters trust and improved interaction with stakeholders. Agriculture employed half of India's workforce and provided 17–18% of the country's GDP. Agriculture and related industries such as animal husbandry, forestry, and fisheries accounted for 15.4% of GDP in 2016 and employed around 31% of the workforce in 2014. The



```
{"intents": [
        {"tag": "greeting",
         "patterns": ["Hi", "How are you", "Is anyone there?", "Hello", "Good day","Namaste"],
         "responses": ["Hello, thanks for visiting", "Good to see you again", "Hi there, how can I help?"],
         "context_set": ""
        },
        {"tag": "goodbye",
         "patterns": ["Bye", "See you later", "Goodbye"],
         "responses": ["See you later, thanks for visiting", "Have a nice day", "Bye! Come back again soon."]
        },
        {"tag": "thanks",
         "patterns": ["Thanks", "Thank you", "That's helpful"],
         "responses": ["Happy to help!", "Any time!", "My pleasure"]
        },
        {"tag": "Soil Testing",
         "patterns": ["TELL ME ABOUT TAKING SOIL SAMPLE ", "How to take soil sample?" , "What is the technique used in taking soil sample"],
         "responses": ["DEEG 6-9 INCH DEEP `V` SHAPE PIT AND TACKING 500 GM SOIL SAMPLE FOR TESTING SOIL", "Dig out a wedge of soil about 6 to 8 inches deep, and set this wedge aside
        },
        {"tag": "Nutrient Management",
         "patterns": ["TELL ME SOMETHING ABOUT NUTRIENT MANAGEMENT ?", "What are some ways to improve nutrients in soil","How to manage nutrients" ],
         "responses": ["SPRAY OF ZINK SULPHATE 5 GRAM +UREA 10 GRAM PER LITER WATER", "SPRAY OF SOLUBLE NPK  18:18:18 GRADE 10 GM/LTR WATER", "SPRAY OF MULTIPLAX (MICRONUTRIENT) 4 ML
        },
        {"tag": "Plant Protection",
         "patterns": ["TELL ME HOW TO CONTROL INSECT IN WHEAT", "What are some ways to control insects in wheat"],
         "responses": ["You can use fumigants like methyl bromide and magnesium or aluminium phosphide"]
        },
        {"tag": "Subsidy",
         "patterns": [ "GIVE ME SOME INFORMATION ON SUBSIDIES","What are some various subsidy data? "],
         "responses": ["Do you need subsidy value for solar plant, farm pond or pipe line ? "],
         "context_set": "subsidyinfo"
        },
        {"tag": "farm pond",
         "patterns": ["Farm pond" ],
         "responses": ["SUBSIDY ON FARM POND = 50% OF TOTAL COST OR 75000/- MAXIMUM"],
         "context_filter": "subsidyinfo"
        },
        {"tag": "solar plant",
         "patterns": ["Solar plant"],
         "responses": ["SUBSIDY ON SOLAR PLANT=70% OF TOTAL COST"],
         "context_filter": "subsidyinfo"
        },
        {"tag": "pipe line",
         "patterns": ["Pipe line"],
```

Figure 1: Dataset format

goal of this project is to create a Chatbot that uses natural language processing to facilitate remote interaction between users/farmers and the agriculture environment. We aim to create a chatbot that can answer basic questions from farmers and give possible agricultural knowledge and solutions. Because this chatbot has been educated in natural language processing, it can learn on its own and improvise responses. The study's target audience is agriculturists or farmers. Their work will be based on the model established. As a consequence, agriculturists will be able to benefit from the study's findings. It has an influence on farmers' livelihoods, helps organizations adopt data-driven agriculture, and fosters trust and improved interaction with stakeholders. The farmers' favorable reaction suggests that conversational intelligence, as a technology supplied via the omnipresent smartphone, can be a useful tool for improving information access in rural areas for those with low literacy and technological expertise.

3 Methodology

3.1 Dataset source and format

The dataset used for this project is taken from data world repository. A multidisciplinary or specialized conversational chatbot is possible. The chatbot's capabilities are influenced by the amount of data utilised to train it. The data is saved as a json file. Tags, patterns, responses, and context are used to organise the data: (.A and Anto, 2013)

Tags: Possible classes of user intention for asking a question.

Patterns: The ways in which users usually ask questions relating to a particular tag.

Responses: Predefined responses for each tag in the dataset from which the model can choose to respond to a particular question.

Context: Contextual words relating to a tag for easy and better classification of what the user intends with their request.

3.2 Workflow diagram

Businesses must understand the workflow of these bots in order to build a chatbot that offers appealing outcomes(Weng, 2019). From the time the chatbot receives a user's query until the time it delivers an answer, the data travels through a number of algorithms that assist the chatbot in comprehending the input. Specifically, most chatbots use several categorization techniques to build up their fundamental architecture Figure 2.

3.3 Data preprocessing

The data has been extracted in json format itself using google colab. Using json.loads() the data is loaded into the system. Google Research's Collaboratory, or "Colab" for short, is a product. Colab is a web-based Python editor that allows anybody to create and run arbitrary Python code. It's notably useful for machine learning, data analysis,

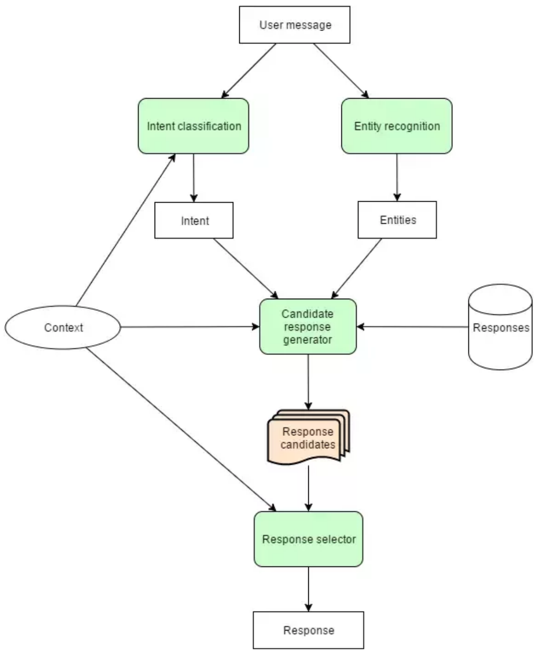


Figure 2: Chatbot Architecture

and teaching. On Google Colab I went with CPU runtime in the first notebook and with the GPU runtime in the second. Then the runtime has been changed to GPU to process the codes faster. We can't just fit a machine learning or deep learning model to the raw text. To begin, we must divide words, handle punctuation and cases, and more in order to prepare the data for modelling. In NLP, cleaning up text data is job specific. For this conversational chatbot we're building, we can do the following. With our intents JSON file loaded, we can now begin to organize our documents, words and classification classes (.A and Anto, 2016).

- Tokenization
- Stemming
- Lemmatization
- Removal of stop words
- Spelling correction
- Normalization
- Removal of punctuation marks
- Creation of training data

1)Tokenization: Tokenization is the process of breaking down a text corpus into constituent words, such as breaking down a phrase, sentence, paragraph, or even an entire text document into smaller units like individual words or terms. A token is the name given to each of these smaller components. Tokenization can be done manually using white space splitting or using specific tools in libraries like NLTK. After tokenization we organized the dataset into words, classes and documents list.

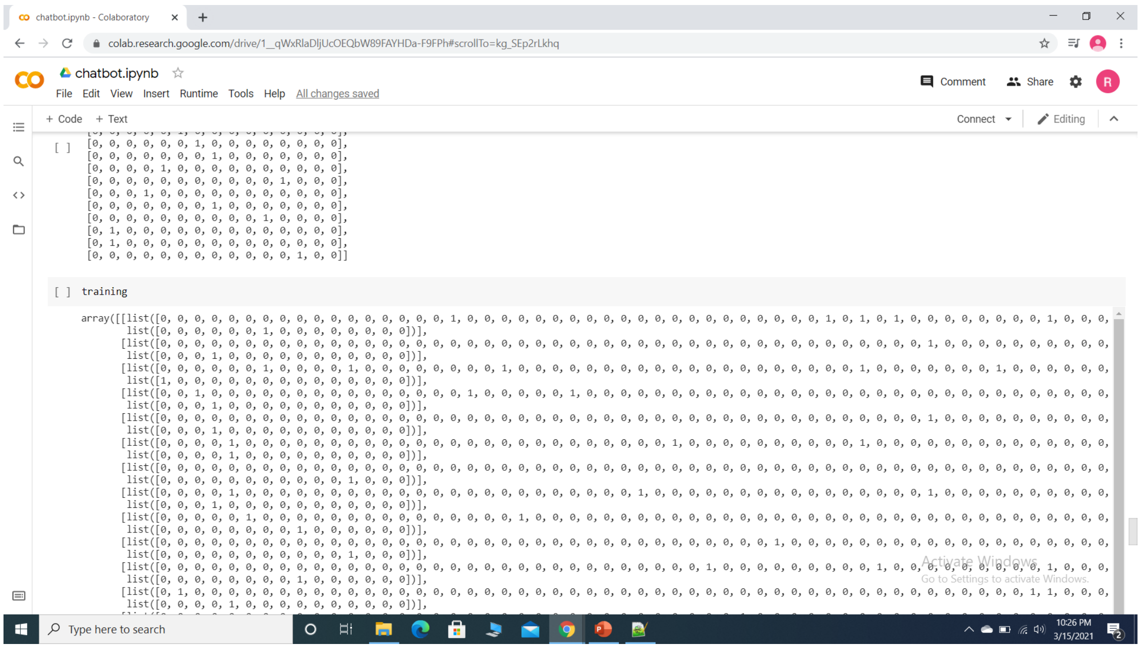
2) Word stemming : The process of creating morphological variations of a root/base word is known as stemming. Stemming algorithms or stemmers are terms used to describe stemming programmes. The terms "chocolates," "chocolatey," and "choco" are reduced to the root word "chocolate," while "retrieval," "retrieved," and "retrieves" are reduced to the stem "retrieve." NLTK has LancasterStemmer class with the help of which we can easily implement Lancaster Stemmer algorithms for the word we want to stem.

3) Lemmatization : Lemmatization is the act of combining a word's several inflected forms into a single item that can be examined. Lemmatization is similar to stemming, except it gives the words context. As a result, it connects words that have similar meanings to one another. The words are morphologically analysed during lemmatization.

4)Removal of stop words: Stop word removal is supported by NLTK, and the list of stop words may be found in the corpus module. To eliminate stop words from a phrase, break your text into words and then check to see if the word is in the NLTK list of stop words.

5) Spelling correction: It is the process of correcting a word's spelling. Because brute force comparisons are extremely time intensive, most spell correction algorithms employ min-edit functions. We need to utilise word lengthening first in order for min-edit features to operate properly. As a result, word lengthening affects our spell correction. Although NLTK lacks a spell-checking module, there are several libraries that can accomplish this function. For this, I'll be utilising the pyspellchecker module. Pyspellchecker is a library for assessing if a word is misspelt and, depending on word frequency, what the likely right spelling is.

6) Text case conversion: We'll use this approach to change the words to lower case or upper case. Although sometimes ignored, one of the simplest and most efficient forms of text preparation is to lowercase all of your text data. It can be used to solve most text mining and NLP issues, and it may


chatbot.ipynb - Colaboratory
colab.research.google.com/drive/1__qWxRlaDljUcOEQbW89FAYHDa-F9FPh#scrollTo=kg_SEp2rLkhq
chatbot.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Connect
Editing
Type here to search
Activate Windows
Go to Settings to activate Windows.
10:26 PM
3/15/2021

```
[ ]  [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
     [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
     [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
     [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
     [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
     [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
     [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
     [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
     [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
     [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]]

[ ]  training

array([[list([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
        list([0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0])],
       [list([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        list([0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])],
       [list([0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
        list([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])],
       [list([0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        list([0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])],
       [list([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        list([0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])],
       [list([0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        list([0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])],
       [list([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        list([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0])],
       [list([0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        list([0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])],
       [list([0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        list([0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0])],
       [list([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        list([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0])],
       [list([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
        list([0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0])],
       [list([0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
        list([0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])],
```

Figure 3: Data after transformation (bag of words)

be very useful when your dataset isn't very huge. It also greatly improves anticipated output consistency.

7) Removal of punctuation marks: Noise reduction is the process of eliminating letters, numbers, and text fragments that might obstruct your text analysis. One of the most important text preparation procedures is noise reduction. It's also quite domain-specific. The framework's replacement duties continue with noise reduction. While the first two major steps of our framework (tokenization and normalisation) could be applied to almost any text chunk or project as-is (barring the decision of which exact implementation to use, or skipping certain optional steps, such as sparse term removal, which does not apply to every project), noise removal is a much more task-specific section of the framework.

8) Creation of training data: Bag of Words (BOW) is a vector space representational model for unstructured text that is one of the simplest. A vector space model is a mathematical model for representing unstructured text (or any other data) as numeric vectors, with each dimension of the vector corresponding to a distinct feature property. Each text document is represented as a numeric vector in the bag of words model, with each dimension being a single word from the corpus and the value being its frequency in the document, occurrence (denoted by 1 or 0), or even weighted values. The term comes from the fact that each text is represented as a 'bag' of its own words, with no concern for word ordering, sequences, or grammar (Jiao, 2020; Bhagwat, 2018).

We need to translate the words into bags of words with arrays containing 0/1. The array length will be equal to vocabulary size, and 1 will be set when a word from the current pattern is located in the given position. Training data — X (pattern converted into array [0,1,0,1..., 0]), Y (intents converted into array [1, 0, 0, 0,...,0], there will be single 1 for intents array).The transformed data is as follows in Figure 3.

3.4 Model implementation

The processed data is used for intent classification using the models so that the model gives promising results. The data is modelled using two deep neural network models that is Multi-Layer Perceptron and Recurrent Neural network model.

1) Multilayer Perceptron (MLP)

A feedforward network having one input layer, one output layer, and at least one hidden layer is known as a multilayer perceptron. Non-linear activation functions, such as the hyperbolic tangent or logistic function, are used to categorize data that is not linear in nature. Every node in the current layer is connected to every node in the following layer, making the network fully connected. After that, the layers are fully connected in a chronological order (input to output), which is known as feed-forward: input -> hidden -> output. Tensorflow's tflearn framework is used to build the MLP network. The fundamental neural network we'll be utilizing is called a Multilayer Perceptron (MLP for short) (Weng, 2019), and it's shown in Figure 4.

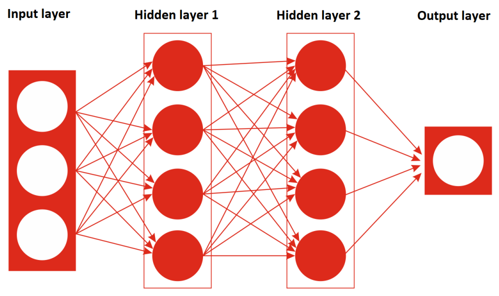


Figure 4: Neural Network – Multilayer Perceptron (MLP)

Certainly, Multilayer Perceptrons have a complex sounding name. However, they are considered one of the most basic neural networks, their design being:

- Input layer – layer "I"
- Hidden layer(s) – layer "H"
- Output layer – layer "O"

2) Recurrent Neural Network

The goal of a Recurrent Neural Network (RNN) is to keep the prior neuron state. This enables the neural network to maintain context and provide output depending on past states. RNNs are ideal for chatbots since preserving context during a discussion is critical to comprehending the user. A RNN model's design is shown in Figure 5.

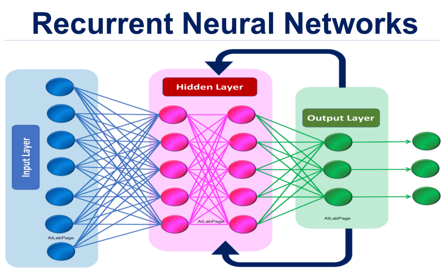


Figure 5: RNN Architecture

We have created the RNN model in following steps:

Step 1: First, we must establish a network model, which will most likely be the Sequential model: the network will be described as a series of layers, each with its own size and activation function that may be customized. The input layer will be the initial layer in these models, and it will need us to determine the size of the input we will be feeding to the network. After this more and more layers can be added and customized until we reach the final output layer (.A and Anto, 2016).

Step 2: After we've created the network's structure in this way, we must compile it, which turns the simple series of layers we've previously specified into a complicated set of matrix operations that determine how the network acts. We must specify the optimization algorithm that will be used to train the network, as well as the loss function that will be minimized, in this section.

Step 3: Once this is done, we can train or fit the network.

Step 4: The network is trained. Now we can use it to make predictions on new data.

Keras is used to create the RNN model. Keras is a high-level open source library for creating neural network models. It was created by François Chollet, a Google Deep Learning researcher. Its fundamental idea is to make the process of creating a neural network, training it, and then utilising it to generate predictions as simple as possible for anyone with a basic understanding of programming, while still allowing developers to fully customize the parameters of the ANN (Weng, 2019).

3.5 Translation of user input data

Once the model is well fitted on the data the next step in creating a chatbot is the creation of a function for translating the user input sentences to the system understandable format. Before we can begin processing intents, we need a way to produce a bag-of-words from user input. The function will process the sentences and converts it into bag of words array. A function has been created that comprises of the major data preprocessing steps and the bag of words array creation steps.

3.6 Response generation

Creating effective chatbots is a difficult task. Creating high-quality natural language replies for chatbots, in particular, is a difficult and time-consuming process that frequently relies on high-quality training data and extensive subject knowledge. As a result, it's critical to include specialists with the necessary subject expertise in the chatbot answer creation process. However, existing tool support for including domain experts in the response creation process is limited, typically limiting users to exchanging disconnected prototypes and spreadsheets. We describe a method in this research that allows chatbot developers to efficiently involve domain experts in the chatbot answer creation process.

The easiest technique, however, still necessitates the creation of some templates in a specific language. It decreases the amount of text input while increasing the number of examples available to train the model. The creation of responses is a crucial stage in the development of a chatbot. I have created response processor using two functions they are classify and response. The intents are classified using the classify function and appropriate response are generated using the response function. Response processor consist of two main functions: Classify for intent classification and response to give appropriate response. I worked on these functions in the view of creating proper responses. Once the chatbot understands the user's message, the next step is to generate a response. One way is to generate a simple static response. Another way is to get a template based on intent and put in some variables. The steps that are involved in creating a response processor is as follows:

- Generate probabilities from the model
- Filter out predictions below a threshold
- Sort by strength of probability
- Return tuple of intent and probability
- If we have a classification then find the matching intent tag
- Loop as long as there are matches to process
- Find a tag matching the first result
- Check if this intent is contextual and applies to this user's conversation
- A random response from the intent

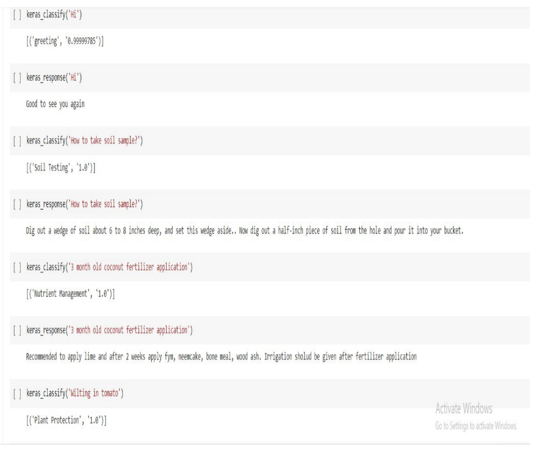
Response() classifies each sentence it receives. Our classifier is fast expanding and uses model.predict(). The model's probabilities are compared to our purpose specifications to provide a list of possible replies. If one or more categories exceed a certain threshold, we check to determine if a tag fits a purpose and proceed accordingly. We'll consider our categorization list as a stack, removing items from it when we discover a good match, or until the stack is empty.

The performance of the response processor is checked using some test data and it gave an excellent performance on predicting the outputs.

4 Result

In this project I have tried implementing Multi-Layer Perceptron model and Recurrent Neural Network models on the dataset. The accuracy given by both the models are comparable, but RNN achieved an accuracy score of 97.83% whereas MLP resulted in an accuracy of 96.97%. The main benefit of RNN over feed forward neural networks is that RNN can represent a collection of records (i.e. time collection), allowing each pattern to be considered to be reliant on the preceding one. In this example, the algorithm is trained using a combination of the knowledge base and behavioral intent scenarios.

The performance of both the models are then evaluated by fitting them for predictions in the response processor. Response processor was designed in such a way that the intent classification is done using the trained models and it produces tuples with class labels and corresponding accuracy of predictions made. The below Figure 6 and Figure 7 shows the performance evaluation of the models in the response processor.

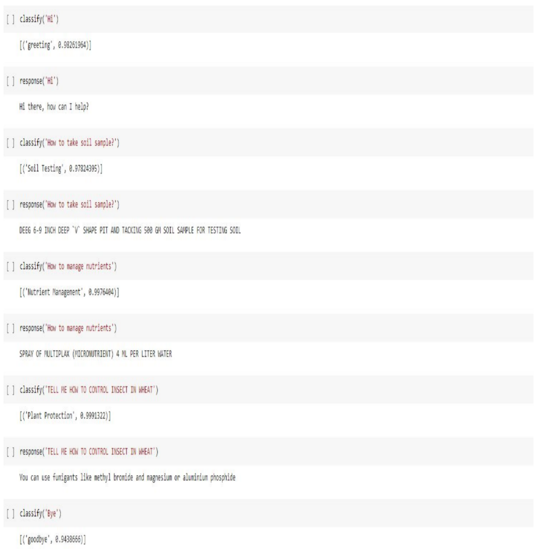


```
[ ] keras_classify('Hi')

    [('greeting', '0.99999785')]

[ ] keras_response('Hi')

    Good to see you again

[ ] keras_classify('How to take soil sample?')

    [('Soil Testing', '1.0')]

[ ] keras_response('How to take soil sample?')

    Dig out a wedge of soil about 6 to 8 inches deep, and set this wedge aside.. Now dig out a half-inch piece of soil from the hole and pour it into your bucket.

[ ] keras_classify('3 month old coconut fertilizer application')

    [('Nutrient Management', '1.0')]

[ ] keras_response('3 month old coconut fertilizer application')

    Recommended to apply lime and after 2 weeks apply fym, neemcake, bone meal, wood ash. Irrigation sholud be given after fertilizer application

[ ] keras_classify('Wilting in tomato')

    [('Plant Protection', '1.0')]
```

Figure 6: Predictions by RNN model.

5 Conclusion

Using natural language technology, this chatbot can help underprivileged areas by answering questions about agriculture, horticulture, and animal husbandry. Through a messaging app, the farmer



```
[ ] classify('Hi')

    [('greeting', 0.98261964)]

[ ] response('Hi')

    Hi there, how can I help?

[ ] classify('How to take soil sample?')

    [('Soil Testing', 0.97824395)]

[ ] response('How to take soil sample?')

    DEEG 6-9 INCH DEEP `V` SHAPE PIT AND TACKING 500 GM SOIL SAMPLE FOR TESTING SOIL

[ ] classify('How to manage nutrients')

    [('Nutrient Management', 0.9976404)]

[ ] response('How to manage nutrients')

    SPRAY OF MULTIPLAX (MICRONUTRIENT) 4 ML PER LITER WATER

[ ] classify('TELL ME HOW TO CONTROL INSECT IN WHEAT')

    [('Plant Protection', 0.9991322)]

[ ] response('TELL ME HOW TO CONTROL INSECT IN WHEAT')

    You can use fumigants like methyl bromide and magnesium or aluminium phosphide

[ ] classify('Bye')

    [('goodbye', 0.9438666)]
```

Figure 7: Predictions by MLP model

will be able to get agricultural information as well as localized information such as current market prices for specific commodities in his/her district and disease management. A farmer can send a direct message to our intelligent answering system and receive a response. Our approach would allow farmers to ask as many questions as they want, at any time, allowing current farming technologies to reach a larger number of farmers faster. Agriculture chatbots play a critical role in the agriculture industry, assisting all farmers and others interested in agricultural operations by analyzing queries and providing relevant information. This Question-Answer system is capable of answering most inquiries without the need for human interaction and with excellent accuracy. This would result in greater human resource use and the avoidance of needless expenditures associated with the establishment of additional contact centres.

Above all, I believe the method aids in the analysis of farmers' mindsets as well as the structure of India's agricultural sector. While the technology provides a safe communication route for farmers, it also aids policymakers in comprehending their wants and concerns. The data analysis also reveals which sectors or seasons demand special attention from farmers. As a result, the decision support system makes effective use of all available resources to address the problem of lack of awareness and knowledge in India's agricultural industry.

Acknowledgments

The authors extend their appreciation to the Deputyship of RESEARCH AND INNOVATION wing of TECHTERN Pvt. Ltd. through SMART-AGRO research and for providing all support for this research work with the project number TTRD-DS-04-2021. This is also an extension of a post-doctoral research program under Kannur University

References

Raji .A and Babu Anto. 2013. Morphological synthesizer a linguistic tool for malayalam verbs. *International Journal of Computational Linguistics and Natural Language Processing*, 2.

Raji .A and Babu Anto. 2016. Knowledge base intelligent query processing. *International Journal of Latest Trends in Engineering and Technology Special Issue SACAIM*, 2:7–12.

Jeremy Beaudry, Alyssa Consigli, Colleen Clark, and Keith J Robinson. 2019. Getting ready for adult healthcare: designing a chatbot to coach adolescents with special health needs through the transitions of care. *Journal of pediatric nursing*, 49:85–91.

Vyas Ajay Bhagwat. 2018. Deep learning for chatbots.

Bob Heller, Mike Proctor, Dean Mah, Lisa Jewell, and Bill Cheung. 2005. Freudbot: An investigation of chatbot technology in distance education. In *EdMedia+ Innovate Learning*, pages 3913–3918. Association for the Advancement of Computing in Education (AACE).

Anran Jiao. 2020. An intelligent chatbot system based on entity extraction using rasa nlu and neural network. In *Journal of Physics: Conference Series*, volume 1487, page 012014. IOP Publishing.

K. K. 2020. Agrochatbot: Understanding the scope of agriculture. *International Journal of Advanced Research in Science Technology (IJARST)*, 07:6.

Rhio Sutoyo, Andry Chowanda, Agnes Kurniati, and Rini Wongso. 2019. Designing an emotionally realistic chatbot framework to enhance its believability with aiml and information states. *Procedia Computer Science*, 157:621–628.

J Weng. 2019. Nlp text preprocessing: A practical guide and template. *Towards Data Science*, 26.

Image2tweet: Datasets in Hindi and English for Generating Tweets from Images

Rishabh Jha[1] **Varshith Kaki** [1] **Varuna Krishna Kolla**[1] **Shubham Bhagat**[1]
Parth Patwa[2] **Amitava Das**[3,4] **Santanu Pal**[3]
[1]Indian Institute of Information Technology Sri City, India
[2]University of California Los Angeles, USA
[3]Wipro AI Labs, India [4]AI Institute, University of South Carolina, USA
[1]{rishabh.j19, vishnusaivarshith.k19, varunakrishna.k19, shubham.b18}@iiits.in
[2]parthpatwa@g.ucla.edu
[3]{amitava.das2, santanu.pal2 }@wipro.com

Abstract

Image Captioning as a task that has seen major updates over time. In recent methods, visual-linguistic grounding of the image-text pair is leveraged. This includes either generating the textual description of the objects and entities present within the image in constrained manner, or generating detailed description of these entities as a paragraph. But there is still a long way to go towards being able to generate text that is not only semantically richer, but also contains real world knowledge in it. This is the motivation behind exploring image2tweet generation through the lens of existing image-captioning approaches. At the same time, there is little research in image captioning in Indian languages like Hindi. In this paper, we release Hindi and English datasets for the task of tweet generation given an image. The aim is to generate a specialized text like a tweet, that is not a direct result of visual-linguistic grounding that is usually leveraged in similar tasks, but conveys a message that factors-in not only the visual content of the image, but also additional real world contextual information associated with the event described within the image as closely as possible. Further, We provide baseline DL models on our data and invite researchers to build more sophisticated systems for the problem.

1 Introduction

Generating a textual description of an image is called image captioning. It can be an easy process for most adults, but for a machine to generate a rich and vivid description is a difficult task. Image captioning requires to recognize the important objects, their attributes and their relationships in an image. It also needs to generate syntactically and semantically correct sentences. This task involves the knowledge of both computer vision and natural language processing.

Image Captioning has been a very popular research area since the last decade. Even before the boom of neural network based techniques people tried various hand crafted features such as Local Binary Patterns (LBP) (Ojala et al., 2000), Scale-Invariant Feature Transform (SIFT) (Lowe, 2004), the Histogram of Oriented Gradients (HOG) (De Marneffe et al., 2006) along with classical ML methods like SVM for Image Captioning. On the other hand, while using neural network based techniques, features are learned automatically from training data and they can handle a large and diverse set of images (Karpathy and Fei-Fei, 2015; Vinyals et al., 2015; Xu et al., 2015). Moreover, the availability of large and new datasets has made the learning-based image captioning an interesting research area. The popular datasets for English Image Captioning are - Flickr30K Dataset (Young et al., 2014), MS COCO (Lin et al., 2014), and Google Conceptual Caption dataset (Sharma et al., 2018). However, there is almost no research of image captioning in Hindi and/or Indian languages.

Image captioning is important for many reasons. For example, they can be used for automatic image indexing. Image indexing is important for Content-Based Image Retrieval (CBIR) and therefore, it can be applied to many areas, including biomedicine, commerce, the military, education, digital libraries, and web searching.

Image2Tweet takes one step ahead of regular image captioning task. It involves generating captions that are not only semantically rich but also contain some real world knowledge

Proceedings of the 18th International Conference on Natural Language Processing, pages 670–676
Silchar, India. December 16 - 19, 2021. ©2021 NLP Association of India (NLPAI)

(Sharma, 2020). The task is that given an image, the machine has to generate a tweet from it. An example is provided in figure 1. Generating this level of detailed tweets requires person identification (Sachin Tendulkar), Object detection (BJP logo) etc.

In this paper, we describe the image2tweet task and release a new dataset for the task and also release a novel hindi dataset to ignite the Image Captioning research for Indian languages.

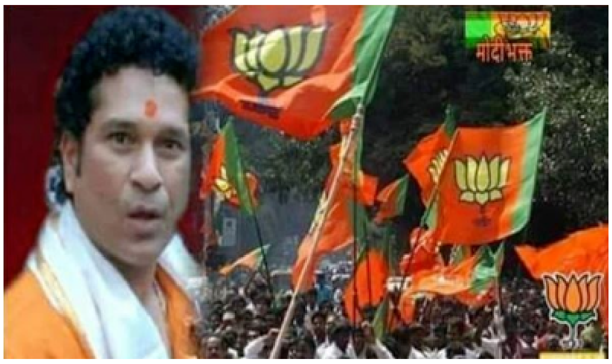


Figure 1:
COCO Style: A man in front of a crowd.
Conceptual Caption: A closeup of a mid-aged man, and a parade.
Expected Image2Tweet: Sachin Tendulkar and BJP parade.

2 Related Work

There are quite a few popular image captioning datasets. Flickr30k (Young et al., 2014) consists of 30K images and each image has 5 captions. COCO (Lin et al., 2014) dataset consists of 330K images and each image has 5 captions. Google Conceptual Caption (Sharma et al., 2018) has approximately 3.3 million images and each image has only one caption. However, such datasets use commonly found images over the web and couple the images with alt-text descriptions. Most of the descriptions use proper nouns (such as *characters, places, locations, organizations, etc.*). Such proper nouns pose some problems because a image captioning model is difficult to learn such fine-grained proper noun inference from the input image pixels. At the same time, there is very little research done on Hindi image captioning. To the best of our knowledge, ours is the first dataset to generate tweet from images and to release a Hindi dataset.

Deep learning methods are the most popular to solve the image captioning task. Jiang et al. (2018) proposed novel Recurrent Fusion Network (RFNet), which exploits complementary information from multiple encoders to tackle image captioning. Xu et al. (2015) propose an encoder-decoder method which incorporate spatial attention mechanism to help the model to determine which regions to focus in an image. Yang et al. (2016) propose a framework called ReviewNet. Zhou et al. (2020) proposed a Unified Vision-Language Pre-Training for Image Captioning which can be easily fine tuned.

Similar to caption generator meme generation has also been a eye-catching task for researchers. the task is to generate memes based on the image. unlike captioning, here in meme generation it has to generate text for multiple persons, if multiple persons are involved in meme image. Kurochkin (2020) released a dataset consisting of 650K meme instances. They applied GPT-2(Radford et al., 2019) model for meme generation and observed that machine generated meme text's are not that engaging as human generated.

3 Task Description

Image Captioning for English is well studied paradigm and researchers have tried various methods like hand crafted features (Ojala et al., 2000; Lowe, 2004; De Marneffe et al., 2006) along with classical ML methods like SVM. During the last decade numerous of Big datasets have been released and quite a few efforts can be noticed but there is a still shortage of works in Indic Languages.

Image2Tweet is a shared task where we move a step forward from image captioning. The task is to generate a tweet like a human/news reporter given an image. We release datasets for two languages - English and Hindi. Figures 2 and 3 show an instance from the English and Hindi data respectively.

3.1 Evaluation Metric

For Image Captioning, most used metrics are n-gram based matching metrics such as BLEU, ROUGE, METEOR, and CIDEr.

Popular Image Captioning datasets like Flickr30k (Young et al., 2014), COCO (Lin et al., 2014), and Google Conceptual Caption

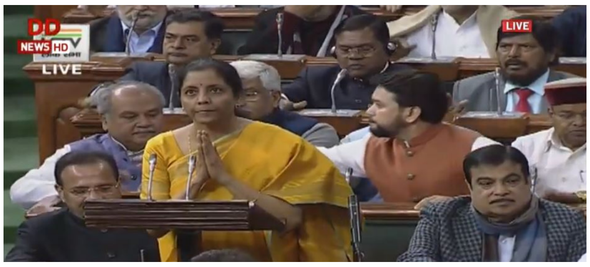


Figure 2: **Tweet**: Finance Minister Nirmala Sitharaman presents the full Budget of the second term of the Narendra Modi government #BudgetSession2020 #BudgetWithTimes #UnionBudget2020

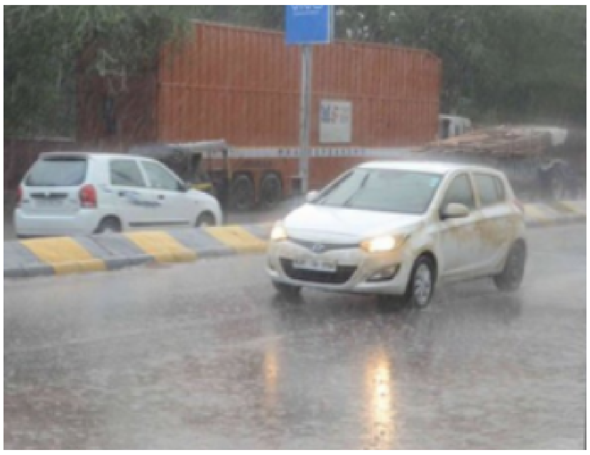


Figure 3: **Tweet**: पिंकसिटी में सुबह से हो रही झमाझम बारिश किसी के लिए राहत तो कहीं आफत #jaipur #Monsoon2017.

(Sharma et al., 2018) provide multiple captions per image, as the same image can be described in many different ways. So, in these datasets, while evaluating they calculate the score between the system generated caption and all the reference captions in the gold data. Now, in our task having multiple tweets for a given image is difficult to collect, and having only one reference tweet will affect the evaluation score.

Since having multiple tweets for an image would be difficult, we assume that similar images may have similar tweets. With this in mind we apply content based similarity match on the collected data and keep all the similar images in one cluster. The released data is preprocessed accordingly, and all the clusters are marked along with image ids. For evaluation, we use CIDEr, where the score will be calculated between system generated tweet vs. all the tweets belong to the similar image cluster provided in the dataset.

4 Dataset

The data consists of image-tweet pairs. We provide 2 dataset - English and Hindi. The Hindi data is collected by crawling tweets from two well known Hindi Newspapers - Dainik Bhaskar and Dainik Jagran. The English data is crawled from the twitter handle of Times of India. We use Twitter API[1] to crawl the tweets. We collect total 70k Tweets for English Image2Tweet and 51K for Hindi Image2Tweet. Table 1 gives the data statistics.

Dataset	**English**	**Hindi**
Training	48792	35701
Validation	10209	7652
Test	10411	7652
Total	69412	51005

Table 1: Train, Validation and Test data split for the English and Hindi datasets.

Figures 4 and 5 show the word clouds of Hindi and English tweets respectively. We observe that most of the words are related to politics and Covid-19.

Clustering is the necessary part of making

[1]`https://developer.twitter.com/en/docs/twitter-api`

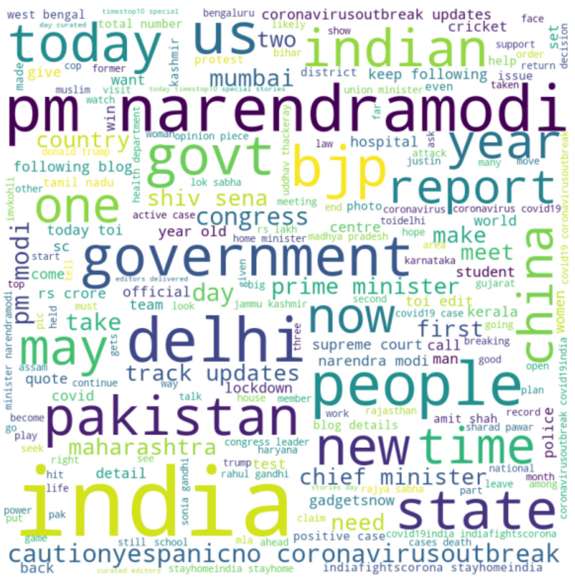


Figure 4: Word cloud of English dataset. Most of the words are related to Politics and Covid-19.

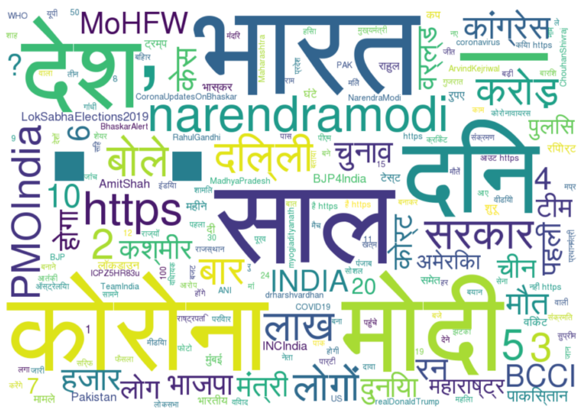
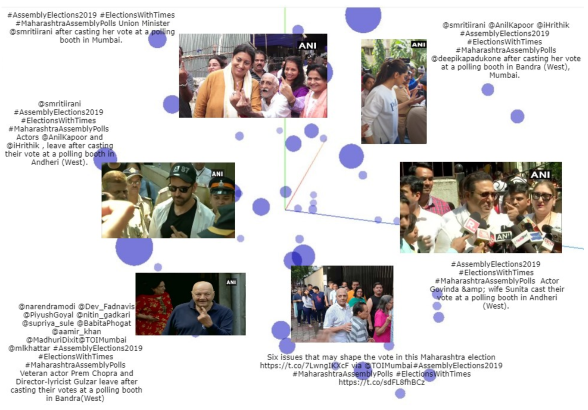


Figure 5: Word cloud of Hindi dataset. Most of the words are related to Politics and Covid-19.



#AssemblyElections2019 #ElectionsWithTimes #MaharashtraAssemblyPolls Union Minister @smritiirani after casting her vote at a polling booth in Mumbai.

@smritiirani @AnilKapoor @iHrithik #AssemblyElections2019 #ElectionsWithTimes #MaharashtraAssemblyPolls @deepikapadukone after casting her vote at a polling booth in Bandra (West), Mumbai.

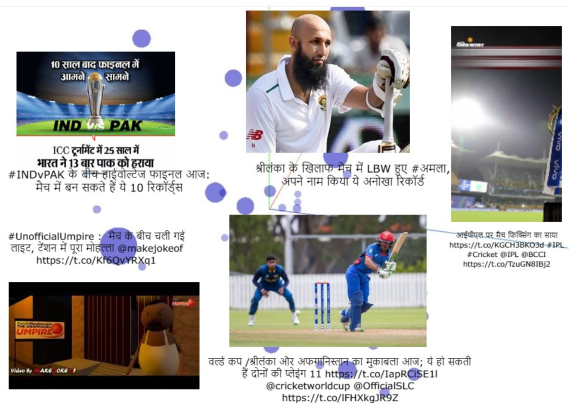
@smritiirani #AssemblyElections2019 #ElectionsWithTimes #MaharashtraAssemblyPolls Actors @AnilKapoor and @iHrithik , leave after casting their vote at a polling booth in Andheri (West).

#AssemblyElections2019 #ElectionsWithTimes #MaharashtraAssemblyPolls Actor Govinda &amp; wife Sunita cast their vote at a polling booth in Andheri (West).

@narendramodi @Dev_Fadnavis @PiyushGoyal @nitin_gadkari @supriya_sule @BabitaPhogat @aamir_khan @MadhuriDixit@TOIMumbai @mlkhattar #AssemblyElections2019 #ElectionsWithTimes #MaharashtraAssemblyPolls Veteran actor Prem Chopra and Director-lyricist Gulzar leave after casting their votes at a polling booth in Bandra(West)

Six issues that may shape the vote in this Maharashtra election https://t.co/7LwngIKXcF via @TOIMumbai#AssemblyElections2019 #MaharashtraAssemblyPolls #ElectionsWithTimes https://t.co/sdFL8fhBCz

Figure 6: An example of a cluster from English data. All the tweet objects are related to political elections in India.



ICC टूर्नामेंट में 25 साल में भारत ने 13 बार पाक को हराया #INDvPAK के बीच हाईवोल्टेज फाइनल आज: मैच में बन सकते हैं ये 10 रिकॉर्ड्स

श्रीलंका के खिलाफ मैच में LBW हुए #अमला, अपने नाम किया ये अनोखा रिकॉर्ड

#UnofficialUmpire : मैच के बीच चली गई लाइट, टेंशन में पूरा मोहल्ला @makejokeof https://t.co/Kf6QvYRXq1

आईपीएल पर मैच फिक्सिंग का साया https://t.co/KGCH3BKO3d #IPL #Cricket @IPL @BCCI https://t.co/TzuGN8IBj2

वर्ल्ड कप /श्रीलंका और अफगानिस्तान का मुकाबला आज; ये हो सकती हैं दोनों की प्लेइंग 11 https://t.co/IapRCiSE1I @cricketworldcup @OfficialSLC https://t.co/lFHXkgJR9Z

Figure 7: An example of a cluster from English data. All the tweet objects are related to cricket.

of the dataset, as mentioned in the previous section. For clustering we first remove unnecessary links, symbols and numbers. However keeping the hashtags and mentions can help in the process of clustering similar tweets (without symbols '@' and hashtags). In the next step we remove the words which doesn't add meaning to sentence, stopwords.

Figure 6 and 7 show and example of English and Hindi cluster respectively. We can see that the tweet objects within a cluster are related/similar to each other. Our aim is to do multimodal clustering, Image+Text, hence we implement an algorithm in which similarity score between tweet object are calculated with every other tweet object and stored in the form of 2D dictionary, each row sorted in reverse order. i^{th} row contains the similarity score of i^{th} tweet object with every other object in reverse order. As for every pair of tweet data there are two entries (dict[i][j] & dict[j][i]), we eliminate that entry which is in lower relative position among those two rows. After that for every row we consider at most 5 element with highest similarity score and combine them to make a cluster group. Hence. each cluster has at most 6 tweet objects (1 tweet object and its 5 neighbors). The formula to calculate similarity between 2 tweet objects is :

$$Sim(i,j) = W1 * textSim(i,j) + W2 * imgSim(i,j)$$

textSim(i,j) function calculates the similarity between the textual part of the tweet object using the weighted average of overlap of unigrams, bigrams and trigrams. imgSim(i,j) function calculates the cosine similarity between the feature vectors of the images extracted using DenseNet (Huang et al., 2017). Overall similarity is just the weighted average of both the similarity, where w1 + w2 = 1 and (w1, w2) ϵ [0,1].

The datasets are available at https://competitions.codalab.org/competitions/35702.

5 Baseline

We develop our baseline using BERT (Devlin et al., 2018) and VGG-19(Simonyan and Zisserman, 2014). The BERT model is pre-trained on whole English Wikipedia and Brown corpus for next sentence prediction objective.

We design is a two branch model (refer figure 8). While training, the image embedding obtained from VGG-19 is passed to one branch and the text is passed to the other branch. In the first branch the image embedding is passed to dense layer. In the second branch the text is sent to BERT tokenizer and its output passed to the pre-trained BERT. Then the output from the last layer of BERT is passed to an LSTM layer which is given as an input to max pooling and to and average polling. The output vectors of max pooling and average pooling are concatenated. After this, we concatenate the outputs from both the branches, and give the concatenated vector as input to an LSTM followed by a dense layer. The output vector of this dense layer is used to generate the words in Tweet.

For training we use Adam optimiser, and train it with a mini-batch size of 32. The learning rate is set to 1e -5. The max caption length is set to 34. While testing, we pass the image vector and the sequence of words generated so far and will predict the next word. Likewise we go on until the end token appears. We use greedy search method to generate the whole tweet.

The baseline code is available at https://github.com/git-rishabh-jha/Image2Tweet.

6 Results

Table 2 shows the results of the baseline system. The results are poor since we use a relatively simple approach to establish the baseline. There is a huge scope of improvement in the results, for which we encourage more innovative approaches.

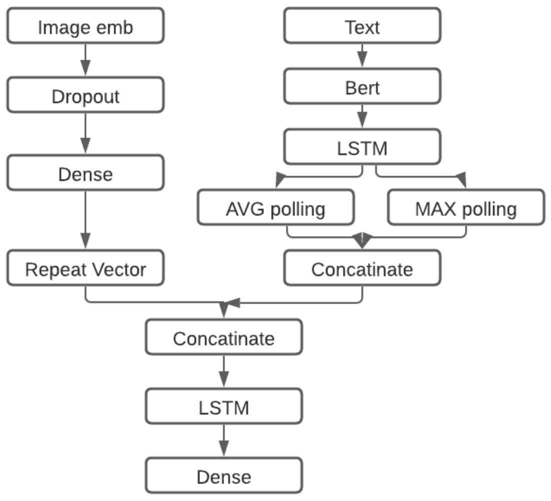


Figure 8: Architecture diagram of baseline model.

Table2 shows the results of the baseline on the image2tweet datasets. The results are poor since we use a relatively simple approach to establish the baseline. There is a huge scope of improvement in the results, for which We encourage more innovative approaches.

Table 3 shows the result of the baseline trained and tested on popular image captioning datasets. The results are much better than on our image2tweet datasets, which shows that image2tweet is a unique and more difficult task than image captioning.

7 Conclusion

In this paper we define the task image2tweet and release datasets in Hindi and English for the task. The English and Hindi datasets consists of 70k and 51k image-tweet pairs respectively. We cluster the similar tweets in our dataset for better evaluation of the system generated tweets. Generated tweets are evaluated using Cider. Further, we provide VGG19 + BERT based baseline systems for our data.

Image2tweet is more difficult than traditional image captioning and we believe it needs further research attention. Future work includes collecting data for more languages, building more complex systems for the task etc.

System	CIDEr	BLEU-4	METEOR	ROUGE
Baseline-English	0.0003	0.02	0.00013	0.00013
Baseline-Hindi	0.0004	0.03	0.00023	0.00023

Table 2: Results of baseline systems on Hindi and English datasets

Dataset	BLEU-4
Flickr30K	23.6
COCO	21.3
Conceptual Captions	20.3

Table 3: Results of baseline systems on popular image captioning datasets.

References

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Lrec*, volume 6, pages 449–454.

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.*

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

Wenhao Jiang, Lin Ma, Yu-Gang Jiang, Wei Liu, and Tong Zhang. 2018. Recurrent fusion network for image captioning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 499–515.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137.

Andrew Kurochkin. 2020. Meme generation for social media audience engagement.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

David G Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. 2000. Gray scale and rotation invariant texture classification with local binary patterns. In *European Conference on Computer Vision*, pages 404–420. Springer.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. 2018. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565.

Shivam Sharma. 2020. Generating tweet-like text from images. where we are...and where we need to be.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR.

Zhilin Yang, Ye Yuan, Yuexin Wu, William W Cohen, and Russ R Salakhutdinov. 2016. Review networks for caption generation. *Advances in neural information processing systems*, 29:2361–2369.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.

Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason Corso, and Jianfeng Gao. 2020. Unified vision-language pre-training for image captioning and vqa. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13041–13049.

Author Index