ICON 2022

International Conference on Natural Language Processing

Proceedings of the Conference

December 15-18, 2022

The ICON organizers gratefully acknowledge the support from the following sponsors.

Gold

SAMSUNG

Adobe

Silver

Google

tcs Research

IIITD | CSE

Infosys Centre for Artificial Intelligence

Bobble AI

TRINKA

Travel and Printing Sponsor

SERB INDIA

Workshop on ICLrNLP Sponsor

Artificial Intelligence

www.elsevier.com/locate/artint

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
209 N. Eighth Street
Stroudsburg, PA 18360
USA
Tel: +1-570-476-8006
Fax: +1-570-476-0860
acl@aclweb.org

ISBN 978-1-959429-38-8

Introduction

There has been a distinct upward trend within NLP and allied areas, primarily since the recent advent of Large Language Models (LLM). The astounding availability of large amounts of information and data has spurred this evolution. This has made developing language technologies for Indian languages quite amenable, with exponential monolingual and multilingual data being constantly added across the web. Similar adoption of of LLM-based NLP solutionsNLP solutions aimed at complex use cases like multilingual chatbots and sentiment analysis for social media discourses is in demand by businesses across industries. Another prominent application that has predominantly captured the technology market space is text-based search in non-Anglophonic languages. An example of increased industrial adoption is accelerating state-of-the-art development in prominent research areas including searching, information extraction, sentiment analysis, and question-answering capabilities for low-resource languages.

The pursuit of solutions to the current existent challenges has led various players across the industry and academia to nurture their research ecosystem right from the funding stage, leading up to tech transfer and adoption. This has facilitated reviewing the tractability of complex use cases that would have been considered a distant dream until a few years ago. Various startups have started venturing into businesses that tread NLP space for applications like multilingual conversation analysis, transcription, and conversational platforms. Similarly, government agencies, institutions, and industry stakeholders have started building consortia-like collaborations that work towards common large-scale objectives involving linguistic and user studies, dataset building, investigation setup design, etc. Leveraging the scope that NLP and allied areas project within the context of Indian research and development, more and more undergraduate and graduate students have started to demonstrate keenness within these areas.

Higher participation, reverberating enthusiasm, and engagement amongst researchers from academia and industry alike have resulted in greater standardization and seamless cross-technology development. The effect can be uniformly seen gaining momentum across various NLP research initiatives, especially across Indian languages. This is also evident in the submissions received as part of ICON-2022, which pertained to knowledge integration, code-mixing, semantic structure, and sentiment analysis within unsupervised and meta-learning frameworks. Recently, advancement in this field has been observed due to the increased accessibility and development of several linguistic resources and rich corpora for Indian languages. This also encourages contemplating language agnostic representation learning and language modeling capabilities. Through such collaborative efforts, it is apt to envision technological advancements that will cater not only to the Indian NLP area but the whole world.

These conference proceedings embody papers selected for presentation in technical sessions of ICON-2022. We thank our excellent reviewers from across different parts of the world, for maintaining the highest of standards in critically assessing the quality of . Out of 101 submissions, a total of 40 papers were accepted. Amongst these, 28 were long papers and 12 were short papers, representing diverse, novel and insightful research findings and encompassing a broad spectrum of topics within NLP and linguistics. In addition, the conference featured multiple engaging presentations – 7 tutorials, 5 workshops, and 1 shared task.

We are grateful to Prof. Tim Baldwin, MBZUAI (Abu Dhabi), Prof. Maria Liakata, Queen Mary University of London, and Prof. Pascale Fung, Hong Kong University of Science Technology for honoring us with their keynote lectures at ICON-2022.

We thank all the area chairs at ICON-2022, especially Pushpak Bhattacharyya, Tirthankar Dasgupta, Sobha L, Vasudeva Varma, Dipankar Das, Monojit Chaudhary, Hema A Murthy, C V Jawahar, Kalika Bali, Sudip Kumar Naskar, Sriparna Saha, Ashwini Vaidya, Kamal Kumar Choudhary, Asif Ekbal, Sukomal Pal, Amitava Das, Manish Srivastava, Vinayak Abrol, N. Kishorjit Singh, Raksha Sharma, Siddhartha Mukherjee, Mayank Singh, Thoudam Doren Singh, Debdoot Mukerjee, for the areas: Syntax and Semantics, Computational Psycholinguistics, QA, Information Extraction, Information Retrieval, and Text Mining, Sentiment Analysis and Emotion Recognition, Language Resources and Evaluation, Speech, Multimodality, Machine Translation, NLP Applications, Machine Learning in NLP, Natural Language Text Generation, along with Doctoral Consortium, Shared Task/Tool Contest and Workshop/Tutorials.

We sincerely express our gratitude to the team members of the Laboratory of Computational Social Systems (LCS2) and the Department of CSE - IIITD, especially, Shivani Kumar, Yash Kumar Atri, Sarah Masud, Aseem Srivastava, Megha Sundriyal, Shivam Sharma, Ayan Sen Gupta, Aayushi Shisodia, and Priti Patwal for their timely assistance in organizing the event at the Indraprastha Institute of Information Technology (IIIT-Delhi) a memorable success. We also thank all the volunteers who assisted us in various activities of the conference. We are grateful to all the researchers, academicians, industry liaisons, and all the participants of ICON-2022 who responded to our call for papers, industry connect outreach, collocated shared tasks and workshop proposals; without whose spirited engagements, the conference would not have been a success. We wholeheartedly thank the reviewers who kindly agreed to review the papers and enabled the quality review ecosystem. We also thank the session chairs for dedicating their valuable time to ICON-2022. We also appreciate the financial support from our sponsors, Adobe (Gold), Samsung Research (Gold), TCS Research (Silver), Trinka AI (Silver), Bobble AI (Silver), Infosys Center of Artificial Intelligence - IIITD (Silver), and Department of CSE - IIITD (Silver) in organizing the in-person conference possible after two years of hiatus. Finally, we thank everyone who participated in ICON-2022 in any possible manner.

Organizing Committee

Patron

Ranjan Bose, IIIT Delhi

Advisory Committee Chair

Rajeev Sangal, IIIT Hyderabad

General Chair

Dipti Misra Sharma, IIIT Hyderabad

Program Chairs

Tanmoy Chakraborty, IIT Delhi
Samar Husain, IIT Delhi

Organizing Committee Chair

Md. Shad Akhtar, IIIT Delhi

Web Chair

Aseem Srivastava, IIIT Delhi

Area Chairs

Amitava Das, Wipro AI
Ashwini Vaidya, IIIT Delhi
Asif Ekbal, IIT Patna
Debdoot Mukerjee, Meesho
CV Jawahar, IIIT Hyderabad
Dipankar Das, Jadavpur University
Hema A Murthy, IIT Madras
Kalika Bali, Microsoft
Kamal Kumar Choudhary, IIT Ropar
Mayank Singh, IIT Gandhinagar
Manish Srivastava, IIIT Hyderabad
Monojit Chaudhary, Microsoft
N. Kishorjit Singh, IIIT Manipur
Pushpak Bhattacharyya, IIT Bombay
Raksha Sharma, IIT Roorkee
Santanu Pal, Wipro AI
Siddhartha Mukherjee, Samsung
Sobha L, Au-KBC
Sriparna Saha, IIT Patna
Sukanta Sen, Zoom

Sukomal Pal, IIT (BHU)
Sudip Kumar Naskar, Jadavpur University
Thoudam Doren Singh, NIT Silchar
Tirthankar Dasgupta, TCS
Vasudeva Varma, IIIT Hyderabad
Vinayak Abrol, IIIT Delhi
Ashutosh Modi, IIT Kanpur

Program Committee

Program Committee

Abdul Waheed, Maharaja Agrasen Instittue of Technology, New Delhi, India
Aditya Shukla, Indian Institute of Technology Patna
Aditya Joshi, Seek
Aisha Aijaz, Indraprastha Institute of Information Technology, Delhi
Amit Sah, South Asian University
Amit Kumar, Tarento Technologies Pvt Limited
Ananya Mukherjee, International Institute of Information Technology, Hyderabad
Anil Vuppala, International Institute of Information Technology Hyderabad
Ankush Agarwal, IIT Bombay
Apoorva Singh, Indian Institute of Technology Patna
Arindam Chatterjee, Wipro AI Research
Arun Chauhan, Indian Institute of Information Technology Dharwad
Aseem Srivastava, IIIT Delhi
Asmita Mukherjee, Indraprastha Institute Of Information Technology
Aswarth Abhilash Dara, Amazon
Atanu Mandal, Jadavpur University
Atharva Kulkarni, Carnegie Mellon University
Ayan Sengupta, IIIT Delhi
Balaji Ganesan, IBM Research
Bharathi B, SSN College of Engineering
Chhavi Sharma, Wipro AI Labs
Debajit Sarma, IIT Guwahati
Debajyoty Banik, Kalinga Institute of Industrial Technology
Deepak Gupta, IIT Patna
Dipanjyoti Paul, IIT Patna
Diptesh Kanojia, University of Surrey
Divya Kumari, IITP
Dr. J.P Singh, NIT Patna
Dr. Nongmeikapam Kishorjit Singh, IIIT Manipur
Dushyant Singh Chauhan, IIT Patna
Fazlourrahman Balouchzahi, Instituto Politécnico Nacional (IPN), Center for Computing Research (CIC)
G Krishna, VNR Vignana Jyothi Institute of Engineering and Technology
Ganeshan Malhotra, BITS Pilani, Goa
Gitanjali Kumari, Indian Institute of Technology Patna
Govind Soni, Jawaharlal Nehru University, Delhi
Harika Abburi, IIIT Hyderabad
Harshit Gupta, IIITH
Hridoy Dutta, University of Cambridge
Kamal Gupta, Indian Institute of Technology Patna
Karish Grover, IIITD
Kartik ., University of California Santa Cruz
Kavita Vemuri, IIIT Hyderabad
Kishore Kashyap, Gauhati University
Koyel Ghosh, Central Institute of Techbology, Kokrajhar
Loitongbam Sanayai Meetei, NIT, Silchar

Mamta Mamta, IIT Patna
Manoj Balaji J, Deloitte
Mauajama Firdaus, University of Alberta
Megha Sundriyal, Indraprastha Institute of Information Technology Delhi
Mohammad Ayyaz Azeem, Riphah International University
Mohd Ansari, Jamia Millia Islamia
Muhammad Abulaish, South Asian University
Muzhaffar Hazman, University of Galway
Nalin Kumar, Charles University
Nihar Sahoo, Indian Institute of Technology, Bombay
Nikesh Garera, Flipkart
P. Radha Krishna, NIT, Warangal
Prashant Kapil, IIT PATNA
Pratik Dutta, Stony Brook University
Premjith B, Amrita Vishwa Vidyapeetham
Priyanshi Gupta, Bharati Vidyapeeth's College of Engineering, Delhi
Prof Shikhar Sarma, Gauhati University
Radhika Mamidi, IIIT Hyderabad
Raghav Jain, IIT Patna
Rajat Pandit, West Bengal State University
Ramakrishna Appicharla, IIT Patna
Ramya Tekumalla, Georgia State University
Ratnesh Joshi, IIT Patna
Rina Kumari, Indian Institute of Technology Patna
Rudramurthy V, IBM Research, India
Sabyasachi Kamila, IIT Patna
Sainik Mahata, Jadavpur University
Sakharam Gawade, IIT Bombay
Sandeep Mathias, Presidency University
Santosh Mishra, IIT Patna
Sarah Masud, IIIT Delhi, India
Sayantan Mitra, IIT Patna
Shaily Desai, PVG's College of Engineering and Technology
Shashwat Vaibhav, IIIT-D
Sheyril Agarwal, BITS Pilani
Shishir Paudel, Kathmandu University
Shivam Sharma, Indraprastha Institute of Information Technology, Delhi
Shivani Kumar, Indraprastha Institute of Information Technology Delhi (IIIT-D), India
Shruti Singh, Indian Institute of Technology Gandhinagar
Simran ., Indraprastha Institute of Information Technology, Delhi
Sohini Roychowdhury, Jadavpur University
Sougata Saha, University at Buffalo
Soumitra Ghosh, Indian Institute of Technology Patna, India
Sourav Kumar, IIIT Hyderabad
Sovan Kumar Sahoo, Indian Institute of Technology Patna
Srinivas Bangalore, Interactions LLC
Sukanta Sen, Indian Institute of Technology Patna
Sumana Biswas, University of Galway
Sumit Kumar, Birla Institute of Technology, Mesra
Sunayana Sitaram, Microsoft Research
Sunil Saumya, Indian Institute of Information Technology Dharwad

Keynote Talk: (Un)fairness in Fairness Evaluation

Tim Baldwin
MBZUAI, Abu Dhabi

Abstract: Natural language processing (NLP) has made truly impressive progress in recent years, and is being deployed in an ever-increasing range of user-facing settings. Accompanied by this progress has been a growing realisation of inequities in the performance of naively-trained NLP models for users of different demographics, with minority groups typically experiencing lower performance levels. In this talk, I will discuss the complexities of the evaluation of model fairness, and how standard evaluation practice has led to unfair/misleading claims in the literature.

Bio: Tim Baldwin is Associate Provost (Academic and Student Affairs) and Head of the Department of Natural Language Processing, Mohamed bin Zayed University of Artificial Intelligence in addition to being a Melbourne Laureate Professor in the School of Computing and Information Systems, The University of Melbourne. Tim completed a BSc(CS/Maths) and BA(Linguistics/Japanese) at The University of Melbourne in 1995, and an MEng(CS) and PhD(CS) at the Tokyo Institute of Technology in 1998 and 2001, respectively. Prior to joining The University of Melbourne in 2004, he was a Senior Research Engineer at the Center for the Study of Language and Information, Stanford University (2001-2004). His research has been funded by organisations including the Australia Research Council, Google, Microsoft, Xerox, ByteDance, SEEK, NTT, and Fujitsu, and has been featured in MIT Tech Review, IEEE Spectrum, The Times, and ABC News. He is the author of over 450 peer-reviewed publications across diverse topics in natural language processing and AI, with around 20,000 citations and an h-index of 66 (Google Scholar), in addition to being an ARC Future Fellow, and the recipient of a number of awards at top conferences.

Keynote Talk: Personalised Longitudinal Natural Language Processing

Maria Liakata
Queen Mary University of London

Abstract: In most of the tasks and models that we have made great progress with in recent years, such as text classification and natural language inference, there isn't a notion of time. However many of these tasks are sensitive to changes and temporality in real world data, especially when pertaining to individuals, their behaviour and their evolution over time. I will present our programme of work on personalised longitudinal natural language processing. This consists in developing natural language processing methods to: (1) represent individuals over time from their language and other heterogenous and multi-modal content (2) capture changes in individuals' behaviour over time (3) generate and evaluate synthetic data from individuals' content over time (4) summarise the progress of an individual over time, incorporating information about changes. I will discuss progress and challenges this far as well as the implications of this programme of work for downstream tasks such as mental health monitoring.

Bio: Maria Liakata is Professor in Natural Language Processing (NLP) at the School of Electronic Engineering and Computer Science, Queen Mary University of London and Honorary Professor at the Department of Computer Science, University of Warwick. She holds a UKRI/EPSRC Turing AI fellowship (2020-2025) on Creating time sensitive sensors from user-generated language and heterogeneous content. The research in this fellowship involves developing new methods for NLP and multi-modal data to allow the creation of longitudinal personalized language monitoring. She is also the PI of projects on language sensing for dementia monitoring & diagnosis, opinion summarisation and rumour verification from social media. At the Alan Turing Institute she founded and co-leads the NLP and data science for mental health special interest groups. She has published over 150 papers on topics including sentiment analysis, semantics, summarisation, rumour verification, resources and evaluation and biomedical NLP. She is action editor for the ACL rolling review and regularly holds senior roles in conference and workshop organisation.

Keynote Talk: Mitigating Risks while Forging Ahead with AI Progress

Pascale Fung
Hong Kong University of Science and Technology

Abstract: Natural language processing (NLP) has made truly impressive progress in recent years, and is being deployed in an ever-increasing range of user-facing settings. Accompanied by this progress has been a growing realisation of inequities in the performance of naively-trained NLP models for users of different demographics, with minority groups typically experiencing lower performance levels. In this talk, I will discuss the complexities of the evaluation of model fairness, and how standard evaluation practice has led to unfair/misleading claims in the literature.

Bio: Pascale Fung is a Chair Professor at the Department of Electronic & Computer Engineering at The Hong Kong University of Science & Technology (HKUST), and a visiting professor at the Central Academy of Fine Arts in Beijing. She is an elected Fellow of the Association for the Advancement of Artificial Intelligence (AAAI) for her significant contributions to the field of conversational AI and to the development of ethical AI principles and algorithms, an elected Fellow of the Association for Computational Linguistics (ACL) for her "significant contributions towards statistical NLP, comparable corpora, and building intelligent systems that can understand and empathize with humans". She is an Fellow of the Institute of Electrical and Electronic Engineers (IEEE) for her "contributions to human-machine interactions" and an elected Fellow of the International Speech Communication Association for "fundamental contributions to the interdisciplinary area of spoken language human-machine interactions". She is the Director of HKUST Centre for AI Research (CAiRE), an interdisciplinary research centre on top of all four schools at HKUST. She co-founded the Human Language Technology Center (HLTC). She is an affiliated faculty with the Robotics Institute and the Big Data Institute at HKUST. She is the founding chair of the Women Faculty Association at HKUST. She is an expert on the Global Future Council, a think tank for the World Economic Forum. She represents HKUST on Partnership on AI to Benefit People and Society. She is on the Board of Governors of the IEEE Signal Processing Society. She is a member of the IEEE Working Group to develop an IEEE standard - Recommended Practice for Organizational Governance of Artificial Intelligence. Her research team has won several best and outstanding paper awards at ACL, ACL and NeurIPS workshops.

Table of Contents

Program

Friday, December 16, 2022

09:00 - 09:30	*Inauguaration*
09:30 - 10:30	*Keynoye 1 - Maria Liakata | Queen Mary University of London*
10:30 - 11:00	*Tea Break*
11:00 - 11:40	*Rockstar Paper 1*
11:40 - 12:20	*Sponsors Sessions*
12:30 - 13:30	*Lunch Break*
14:00 - 15:00	*Keynote 2 - Pascale Fung | Hong Kong University of Science & Technology*
14:00 - 15:00	*Keynote 2 - Pascale Fung | Hong Kong University of Science & Technology*
15:00 - 15:20	*Tea Break*
15:20 - 16:20	*Technical Session - I*
15:20 - 16:20	*Technical Session - II*
15:20 - 16:20	*Technical Session - II*
15:20 - 16:20	*Technical Session - IV*
16:30 - 17:30	*Annual NLPAI Meet*
18:30 - 21:00	*Cultural Night & Conference Dinner*
14:00 - 15:00	*Technical Session - VI*

Friday, December 16, 2022 (continued)

Saturday, December 17, 2022

09:00 - 10:00	*Keynote 3 - Tim Baldwin | University of Melbourne*
10:00 - 10:40	*Rockstar paper 2*
10:40 - 11:00	*Tea Break*
11:00 - 12:00	*Panel Discussion on 'In the era of Transformers, would the classical NLP be at risk?'*
12:00 - 12:20	*Sponsor Session*
12:30 - 13:30	*Lunch Break*
14:00 - 15:00	*Technical Session - V*
14:00 - 15:00	*Technical Session - VII*
14:00 - 15:00	*Technical Session - VIII*
15:00 - 15:20	*Tea Break*
15:20 - 16:05	*Technical Session - IX*
15:20 - 16:20	*Technical Session - X*
15:20 - 16:05	*Technical Session - XI*
16:30 - 17:30	*Valedictory*

EdgeGraph: Revisiting Statistical Measures for Language Independent Keyphrase Extraction Leveraging on Bi-grams

Amit Kumar Gupta,
Manipal University Jaipur,
Rajasthan, India.
dramitkumargupta1983@gmail.com

Muskan Garg
Mayo Clinic,
Rochester, MN, USA.
muskanphd@gmail.com

Abstract

The NLP research community resort conventional Word Co-occurrence Network (WCN) for keyphrase extraction using random walk sampling mechanism such as PageRank algorithm to identify candidate words/ phrases. We argue that the nature of WCN is a path-based network and does not follow a *core-periphery structure* as observed in web-page linking network. Thus, the language networks leveraging on bi-grams may represent better semantics for keyphrase extraction using random walk. In this work, we use bi-gram as a node and adjacent bi-grams are linked together to generate an EdgeGraph. We validate our method over four publicly available dataset to demonstrate the effectiveness of our simple yet effective language network and our extensive experiments show that random walk over EdgeGraph representation performs better than conventional WCN. We make our codes and supplementary materials available over Github[1].

1 Introduction

The language network is a textual representation of documents in the shape of a graph to exploit the best features as their characteristics. With recent developments in statistical keyphrase extraction, language network plays a pivotal role in identifying underlying patterns among words, phrases or sentences (Garg, 2021). The research community maps these patterns using the network properties as the structural properties of language networks has gained much attention in recent years (Lu et al., 2021). Existing literature contains substantial studies over the structural properties for different languages (Vera and Palma, 2021) and different domains (Garg and Kumar, 2020; Quispe et al., 2021) resulting into development of real-time language independent and domain-specific techniques, respectively.

[1]https://github.com/drmuskangarg/EdgeGraph

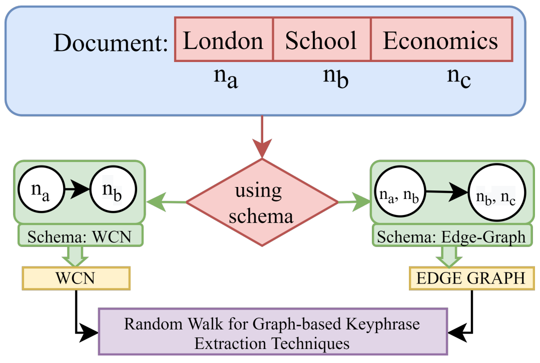


Figure 1: Overview of the proposed work

We further use the structural properties in modeling the dynamics of evolving language networks for downstream NLP applications, for instance, the Dynamic Heartbeart Graph (DHG) for event detection on Twitter (Saeed et al., 2019); and tracking the dynamics of co-word networks for emerging topics (Huang et al., 2021; Katsurai, 2017). An essential element for these graph-based topic detection and tracking applications is keyphrase extraction.

The conventional WCN is established as a benchmark representation of textual documents for random walk based keyphrase extraction(Kazemi et al., 2020; Campos et al., 2020). The random walk sampling is characterized by stochastic movement of several iterations over a network for redistributing weights to nodes. This concept of random walk was initially introduced for *web-page linking* due to the core-periphery structure (Getoor and Diehl, 2005) of the World Wide Web (WWW) connectivity. However, we observe that:

1. The WCN has significant bias towards the node which represents frequently occurring words irrespective of its context.

2. In a WCN, the edge-weight gives better insights than a node degree (Garg and Kumar, 2018a) which shows the importance of bi-gram in a language network .

3. The WCN does not support the core-periphery structure like web-page linking which is an important property for the PageRank algorithm.

In this work, we study a significance of replacing WCN with EdgeGraph for random walk based GKET. The overview of our proposed approach is shown in the Fig. 1. The major contributions of this research are:

1. We propose the EdgeGraph, a graph-based textual representation to increase the information in every node and accommodate edge-distribution property.
2. We use four different publicly available text collections for keyphrase extraction to validate the EdgeGraph over WCN.
3. The statistical studies validates the effectiveness of EdgeGraph over the WCN for English dataset with medium-sized documents.

2 BACKGROUND AND RELATED WORK

The automatic keyword extraction techniques are classified into the *structured* and *unstructured* algorithms. The supervised keyword extraction is not reliable for ever-changing and dynamically evolving information (Florescu and Caragea, 2017). The unsupervised algorithms are either *statistical* or *graph-based*. A well-studied approach of supervised algorithms is graph-based keyphrase extraction.

2.1 Evolution of GKET

The PageRank algorithm (Page et al., 1999) is used for random walk sampling over *web-page linkings*. The TextRank uses PageRank algorithm and establishes itself as the first and one of the most promising random walk based GKET (Mihalcea and Tarau, 2004; Zhang et al., 2020) for textual documents. The extended version of TextRank is biased towards the node scores but explainable and is known as the Biased TextRank (Kazemi et al., 2020). The recent empirical study of TextRank (Zhang et al., 2020) shows the effectiveness of *graph-based keyphrase extraction*. PositionRank is another keyphrase extraction technique in which the position of a token plays a pivotal role (Florescu and Caragea, 2017) in identifying the candidate phrases.

The NErank (Bellaachia and Al-Dhelaan, 2012) is proposed for short-text data using *the node score* and *the edge score* over a WCN. Other than the random walk, some of the path-breaking structural GKET are *degree centrality* (Abilhoa and De Castro, 2014), *selectivity based keyword extraction* (Beliga et al., 2016), *k-core decomposition* (Tixier et al., 2016), and *keyword extraction using collective node weight* (Biswas et al., 2018) which are based on network science metrics/ models and are beyond the scope of this study. In future, the effectiveness of EdgeGraph can be studied for these structural GKET.

2.2 Historical Perspective of WCN

Graph theory has paved the path to explore language networks evolved from textual documents (Choudhury et al., 2010). The structural properties for this language network are *scale-free networks, small world property, hierarchical organization, assortativity and spectral distribution* which are studied for the WCN evolved from Chinese and English language (Liang et al., 2009), Microblogs (Garg and Kumar, 2018b), and 12 other Slavic languages (Liu and Cong, 2013). The WCN follows the *small-world property* and is *disassortative* in nature. The *eigenvalues and the spectral distribution* helps in understanding the vibrations in the linear system of language networks (Liang, 2017).

2.3 Research Gap

The semantic studies for keyword extraction techniques use Wikipedia (Wan and Xiao, 2008a), topical ranking (Awan and Beg, 2021; Bougouin et al., 2013; Boudin, 2018), and semantic connectivity (Duari and Bhatnagar, 2019). Different text-representation for semantic GKET (Osman and Barukub, 2020) are Large-scale Information Network Embedding (LINE) (Tang et al., 2015) and Context Aware Graph (CAG) (Duari and Bhatnagar, 2019). CAG incorporates the context set by two consecutive sentences by linking co-occurring words together. (Duari and Bhatnagar, 2019) use CAG for keyword extraction to eliminate the need of integer-sized sliding window parameters. Variations in weighted and unweighted adjacency matrix (Papagiannopoulou et al., 2021) and revisting this approach in literature (Ushio et al., 2021) shows that there is no existing study for variation in the text-representation with path-based network of words.

3 OUR APPROACH

In this manuscript, we propose a variation in the graph-based text representation. We find *candidate phrases* which seems to be capable to being identified as keyphrases using random walk. In this section, we discuss EdgeGraph representation.

3.1 Problem Formulation

Consider a set of pre-processed documents D as $D = d_1, d_2, d_3, \dots d_k$ where d_i is the i^{th} document. In a document d_i, the sequence of tokens is $t_{i,1}$, $t_{i,2}$, $t_{i,3}$, $\dots$ $t_{i,z}$, where z is the number of tokens in a document. Every token is considered as a node $t_{i,j}$ where i is the position of a document d_i and j is the position of the token in that document d_i. The total number of nodes are m and m' which varies and represents the unique number of tokens for the WCN and EdgeGraph, respectively. We use the token $(t_{i,j}, t_{i,j+1})$ as a node (n_a) in the graph for further simplification.

Definition 1: Word Co-occurrence Network (WCN): The existing WCN is a graph G of words where edges are added as (n_a, n_b). The word adjacency matrix A is created by using the co-occurrence (n_a, n_b) where the first word of the tuple (n_a, n_b) is taken as the row index and the latter word is taken as the column index in the matrix. The adjacency matrix is used to generate a WCN which is mapped as $m * m$ matrix for m: the total number of nodes in the WCN. Thus, the Graph G contains m nodes and every edge is represented as (n_a, n_b).

Definition 2: EdgeGraph: We build EdgeGraph E_G from a set of textual documents D where we map every document d_i in a graph of adjacent bi-grams. Considering a sequence n_a, n_b, n_c, the edge of a graph is the link which exists between the node (n_a, n_b) and the node (n_b, n_c) of the graph E_G. We use the bi-gram (n_a, n_b) as the node. The word adjacency matrix A' is created by using the co-occurrence $((n_a, n_b), (n_b, n_c))$ where the first node of the tuple $((n_a, n_b), (n_b, n_c))$ is taken as the row index and the latter node is taken as the column index in the matrix. The adjacency matrix is used to generate a WCN which is mapped as $m' * m'$ matrix for m': the total number of nodes in the EdgeGraph.

Given the above settings, our task is to study random walk based GKET over WCN and EdgeGraph.

Table 1: Structure of two different graph-based text representations: WCN and EdgeGraph

Graph	#Nodes	#Edges	Node: Edge	Avg Node Degree	Avg Edge Weight
WCN	581	1188	0.49	4.09	1.22
EdgeGraph	1195	1291	0.93	2.16	1.06

3.2 Problem Statement

In the WCN, the PageRank $(PR((n_b); t_q))$ of any node n_b at the time t_q depends upon the PageRank $(PR((n_a); t_{q-1}))$ of the predecessor neighbours n_a of the node n_b. The idea behind this research work is to emphasise the importance of bi-gram connectivity in language network in-place of uni-grams. Thus, the PageRank, for any bi-gram $PR((n_b, n_c); t_q)$ at the time t_q depends upon the PageRank $(PR((n_a, n_b); t_{q-1}))$ of the predecessor neighbours (n_a, n_b) of the node (n_b, n_c). Thus, the PageRank for E_G is represented as PR_G as shown in Equation 1.

$$PR_G(n_b, n_c) = \frac{1-d}{m'} + d \sum_{u \epsilon M(n_a, n_b))} \frac{PR_G(n_a, n_b)}{out(n_a, n_b)} \tag{1}$$

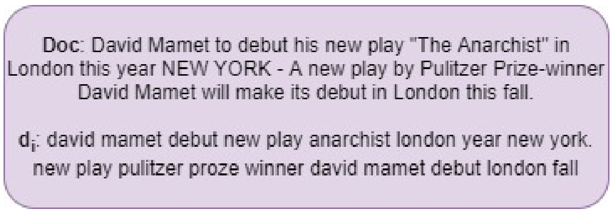
where d is the damping factor, $M(n_a, n_b)$ is a node in the set of node (bi-gram) which are directly linked to the node (n_b, n_c), and m' is total number of nodes in the EdgeGraph E_G, The EdgeGraph is evolved from $m' * m'$ adjacency matrix. The PageRank algorithm is used for random walk in the WCN (PR) and the EdgeGraph (PR_G) representation.

3.3 Working Instances

The proposed work is demonstrated over four different publicly available dataset. We use the text collection *500N-KPCrowd-v1.1* to discuss two types of working instances in this section. The first example differentiates the characteristics of the WCN and the EdgeGraph over one of the documents of *500-KP-Crowd-v1.1* dataset. The second example demonstrates the graph-based text representation of a short-text document of the dataset *500-KP-Crowd-v1.1* as WCN and EdgeGraph.

3.3.1 Example 1: Characteristics of the graph-based text representations

The nature of WCN and EdgeGraph differentiates due to uni-gram and bi-gram adjacency, respectively as shown in Table 1. The number of unique nodes (n_a) in the WCN is lesser than number of



Doc: David Mamet to debut his new play "The Anarchist" in London this year NEW YORK - A new play by Pulitzer Prize-winner David Mamet will make its debut in London this fall.

d_i: david mamet debut new play anarchist london year new york. new play pulitzer proze winner david mamet debut london fall

Figure 2: The working instance document (Doc) and its preprocessed version d_i

Table 2: Indexing of tokens for WCN evolving from the working instance d_i

Index	**Word**	**Index**	**Word**
t_0	david	t_7	year
t_1	mamet	t_8	york
t_2	debut	t_9	pulitzer
t_3	new	t_{10}	prize
t_4	play	t_{11}	winner
t_5	anarchist	t_{12}	fall
t_6	london		

nodes (n_a, n_b) in the EdgeGraph because the neighbour of a word in every node may vary and unlike WCN one word may appear in more than one node in text representation of the same document. This repetition preserves the contextual difference among words with each other. The repetition of bi-grams is very limited in EdgeGraph and thus, the *node to edge ratio* is close to 1 and the *average node degree* is reduced. There is slight *increase and decrease* in *the number of edges and average edge weight*, respectively. If the number of nodes are almost doubled and there is slight increase in the number of edges; the density of the network reduces. As a result, fewer nodes with significant bi-gram are emphasized.

3.3.2 Example 2: Graph-based text representation

Consider an example of a document (Doc) which is pre-processed to obtain the document d_i as shown in Fig. 2. We index the uni-gram and bi-gram as nodes to generate the graph-based textual representation of a document d_i. The indexing of tokens are different for the WCN and the EdgeGraph as shown in Table 2 and Table 3, respectively. The WCN and EdgeGraph are generated using these indexing tables as shown in Fig. 3.

On investigating the connections of a network, we found that the important bi-gram lexical sequence is preserved in EdgeGraph and not in the WCN, for instance, *new york* and *new play* are contextually different but the word *new* is connecting both *play* and *york* in the WCN. The words *play*

Table 3: Indexing of tokens for EdgeGraph evolving from the working instance d_i

Index	**Word**	**Index**	**Word**
t_{G0}	David Mamet	t_{G8}	New York
t_{G1}	Mamet debut	t_{G9}	play pulitzer
t_{G2}	debut new	t_{G10}	pulitzer prize
t_{G3}	new play	t_{G11}	prize winner
t_{G4}	play anarchist	t_{G12}	winner David
t_{G5}	Anarchist London	t_{G13}	debut London
t_{G6}	London year	t_{G14}	London fall
t_{G7}	year new		

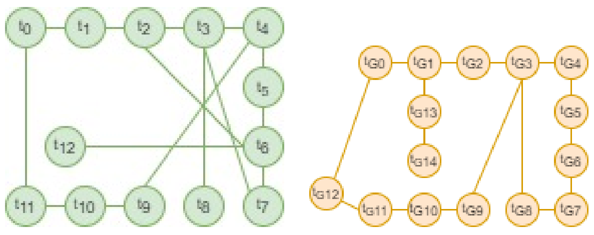


Figure 3: The WCN (left) and EdgeGraph (right) evolved from the document d_i, a working instance in Example 2

and *york* have different dictionary meanings and their connection does not make sense. However, in EdgeGraph, two different nodes preserve these bi-grams as the node *(new york)* and the node *(new play)*. The random walk over WCN may emphasise frequently used but unimportant words like *new* which alone does not make much sense. The EdgeGraph gives importance to meaningful bi-gram like *David Mamet*, *new play*, *new york* which make sense together. If a tuple (a,b) and (b,c) are retrieved, we combine them to form (a, b, c) and thus, n-gram keyphrases are obtained.

4 EXPERIMENTS AND EVALUATION

We perform the experiments with TextRank (Mihalcea and Tarau, 2004), SingleRank (Wan and Xiao, 2008b), PositionRank (Florescu and Caragea, 2017), and NErank (Bellaachia and Al-Dhelaan, 2012) over publicly available text collections. In this section, we discuss the characteristics of datasets, the experimental setup, performance evaluation and statistical significance of the proposed textual representation over the baseline.

4.1 Datasets

To test and validate the robustness of EdgeGraph over WCN, experimental results are carried out for four different datasets whose characteristics are given in Table 4. The average number of tokens per document varies from 20 to 500. The annotated

Table 4: Characteristics of the dataset used for experiments and evaluation of keyphrase extraction

Dataset	Language	Type of Doc	Domain	#Docs	#Tokens/ doc
110-PT-BN-KP	PT	News	Misc.	110	304.00
500N-KP Crowd-v1.1	EN	News	Misc.	500	408.33
pak2018	PL	Abstract	Misc.	50	97.36
wiki20	EN	Research Report	Comp. Science	20	6177.65

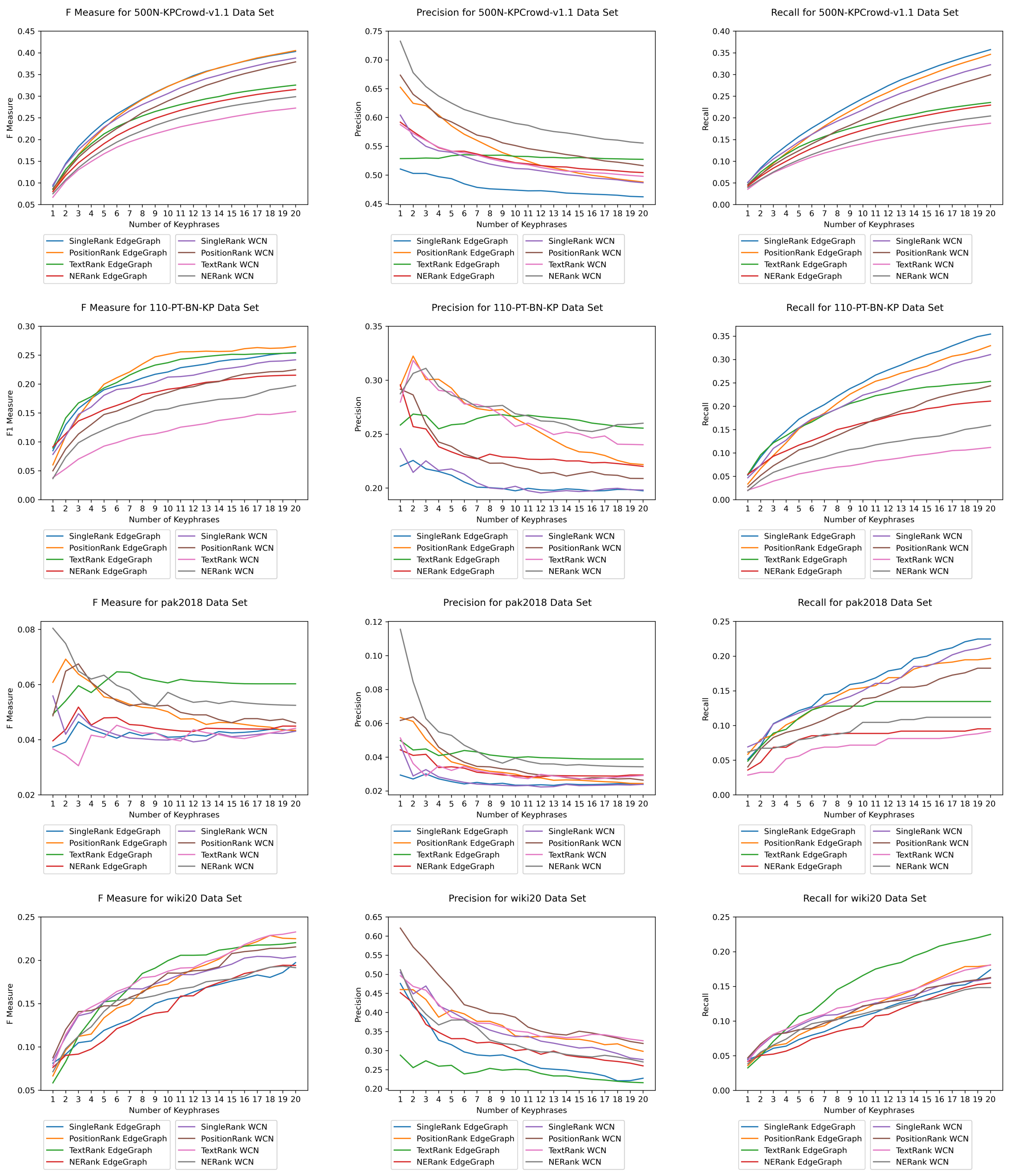


Figure 4: Performance evaluation of the random walk based GKET over WCN and EdgeGraph representation of medium-sized text for varying values of k: F-measure, Precision and Recall.

data is one of the major reasons behind variation in the resulting values of performance evaluation measures due to its subjectivity. This variation is not a potential constraint in this research work as the performance is comparative. We use four different text collections for this study: 110-PT-BN-KP (Marujo et al., 2013), 500N-KP-Crowd-v1.1 (Marujo et al., 2013), pak2018 (Campos et al., 2018), and wiki20 (Medelyan et al., 2008). Three out four dataset contains few lines of text (containing less than 500 words) to display *news* and *abstract* about miscellaneous data in three different languages. These few lines of text are different from short-text and long textual documents and thus, are termed as *medium-sized text*. The dataset *wiki2020* is in the English language which contains the *research paper* in which there are more than 4000 words in each document. We use these characteristics to categorically study the evaluation of results.

4.2 Experimental Setup

The experimental setup for this research work comprises the hardware requirements of CPU @ 2.90 GHz with Intel Core i7-7500 CPU over 64-bit Operating System having 8.00 GB RAM. We use the software of *Python 3* with library modules of *networkx* for graphical analysis, *NLTK* for text processing, *pandas* to handle the data, *matplotlib* for graph plot, and many other relevant modules.

We implement the baselines by using existing modules[2] which are further modified to incorporate the settings for EdgeGraph. We use the default parameter settings of random walk based GKET which are available in the existing implementation. The existing random walk based GKET use varying values of the sliding window parameter to generate the WCN. The most widely used value of sliding window parameter is 2 (Mihalcea and Tarau, 2004; Florescu and Caragea, 2017). The value of the damping factor d is set as 0.85 and the number of iterations are 1000. The network is converged with error rate $\epsilon < 0.01$.

As the results are comparative, we use *student's t-test* to measure the statistical significance of the results. The *Microsoft Office package* is used for the results obtained in (.csv) format to test and validate the robustness of the EdgeGraph for its statistical significance.

[2]https://github.com/boudinfl/pke

4.3 Performance Evaluation

We evaluate the performances using *Precision*, *Recall* and *F-measure* for the varying values of k where k is the number of top ranked keyphrases. All the unique tokens in extracted keyphrases are taken as the set of *extracted words*, and the tokens obtained from the ground-truth keyphrases are taken as the set of *reference words*. The performance is evaluated over these two lists: *extracted words* and *reference words* for increasing values of k over the WCN and the EdgeGraph on every dataset. The results for datasets with medium-sized text and datasets with long-text are shown in Fig. 4.

For English datasets, recall grows more steeply than non-English datasets with increasing value of k. Irrespective of language, recall shows clear improvements over EdgeGraph representation for a higher value of k as shown in Fig. 4. Precision decreases with increase in the value of k. Variation around average value of precision is lesser for medium-sized text than for long-text datasets because the probability of identifying appropriate keyphrases decreases in long-text documents due to reduced probability with large number of tokens. It is interesting to note that precision for TextRank on EdgeGraph remains constant for varying values of k.

4.4 Time Complexity

Since there is no change in the algorithm for random walk based GKET, the time complexity remains same. However, the number of nodes and the information in these nodes is increased. Also, the node to edge ratio decreases which makes the graph sparse. As the random walk is based on the Markov decision process and transition probability is increased due to change in node degree distribution.

4.5 Improvements with EdgeGraph

The experimental results are shown in Table 5. The resulting values of EdgeGraph in bold digit indicates the improvement. The datasets containing medium-sized text in which the number words are less than 500 shows better F-measure improvement over the datasets containing long text. Further, the dataset with English language shows improvement over recall in more than 90% of the cases. However, the resulting values for precision are compromised due to extraction of huge amount of data as ev-

Table 5: Results obtained for random walk based GKET over four different keyphrase extraction datasets using the WCN and the EdgeGraph text representations for $k = 20$.

Dataset 500N-KPC						
Methods	Recall		Precision		F Measure	
	WCN	EdgeGraph	WCN	EdgeGraph	WCN	EdgeGraph
Text Rank	0.1875	**0.2354**	0.4977	**0.5272**	0.2724	**0.3255**
NE Rank	0.2042	**0.2253**	0.5554	0.5042	0.2987	**0.3152**
Position Rank	0.2994	**0.3464**	0.5163	0.4878	0.3790	**0.4051**
Single Rank	0.3224	**0.3573**	0.4865	0.4621	0.3878	**0.4030**
Dataset PAK 2018						
Methods	Recall		Precision		F Measure	
	WCN	EdgeGraph	WCN	EdgeGraph	WCN	EdgeGraph 33
Text Rank	0.0913	**0.1345**	0.0291	**0.0388**	0.0441	**0.0602**
NE Rank	0.1118	0.0951	0.0342	0.0294	0.0524	0.0449
Position Rank	0.1825	**0.1966**	0.0263	0.0244	0.0461	0.0434
Single Rank	0.2166	**0.2246**	0.0239	**0.0239**	0.0430	**0.0431**
Dataset PT BN KP						
Methods	Recall		Precision		F Measure	
	WCN	EdgeGraph	WCN	EdgeGraph	WCN	EdgeGraph
Text Rank	0.2401	**0.2555**	0.2402	**0.2555**	0.2543	**0.2638**
NE Rank	0.2601	0.2201	0.2601	0.2201	0.1973	**0.2153**
Position Rank	0.2088	**0.2217**	0.2088	**0.2217**	0.2249	**0.2651**
Single Rank	0.1982	0.1974	0.1982	0.1974	0.2419	**0.2535**
Dataset WIKI 20						
Methods	Recall		Precision		F Measure	
	WCN	EdgeGraph	WCN	EdgeGraph	WCN	EdgeGraph
Text Rank	0.1809	**0.2249**	0.3258	0.2159	0.2326	0.2249
NE Rank	0.1482	**0.1547**	0.2703	0.2599	0.1914	**0.1940**
Position Rank	0.1627	**0.1805**	0.3186	0.2981	0.2154	**0.2249**
Single Rank	0.1617	**0.1740**	0.2765	0.2276	0.2041	0.1972

Table 6: Statistical Significance for different keyphrase Extraction over WCN and EdgeGraph.

Dataset 500N-KPC						
Methods	Recall		Precision		F Measure	
	t_test	p_value	t_test	p_value	t_test	p_value
Text Rank	8.8480	**3.64E-08**	-11.3725	6.38E-10	10.5934	**2.06E-09**
NE Rank	16.1771	**1.45E-12**	0.8804	***0.3896***	21.4380	**8.98E-15**
Position Rank	12.1549	**2.09E-10**	-14.8203	6.80E-12	20.5013	**2.03E-14**
Single Rank	9.2492	**1.82E-08**	-16.0827	1.61E-12	12.9572	**7.01E-11**
Dataset PAK 2018						
Methods	Recall		Precision		F Measure	
	t_test	p_value	t_test	p_value	t_test	p_value
Text Rank	4.0972	**0.00061**	-0.8953	*0.3817*	-0.4678	*0.6452*
NE Rank	23.8044	**1.31E-15**	13.5307	**3.33E-11**	27.1391	**1.17E-16**
Position Rank	-5.7231	1.62E-05	-4.0292	0.0007	-7.1178	9.08E-07
Single Rank	3.9588	**0.00084**	-1.3992	*0.1778*	-0.3888	*0.7016*
Dataset PT BN KP						
Methods	Recall		Precision		F Measure	
	t_test	p_value	t_test	p_value	t_test	p_value
Text Rank	11.3315	**6.78E-10**	-1.1464	*0.2658*	16.6051	**9.09E-13**
NE Rank	17.2039	**4.83E-13**	-0.7460	*0.4647*	30.9321	**1.02E-17**
Position Rank	22.7884	**2.93E-15**	-12.3763	1.54E-10	19.9476	**3.34E-14**
Single Rank	8.3208	**9.30E-08**	1.4777	***0.1558***	9.008	**2.75E-08**
Dataset WIKI 20						
Methods	Recall		Precision		F Measure	
	t_test	p_value	t_test	p_value	t_test	p_value
Text Rank	-5.0775	6.69E-05	-17.7666	2.71E-13	-12.0685	2.35E-10
NE Rank	5.5577	**2.32E-05**	-16.2917	1.28E-12	-1.1878	*0.2495*
Position Rank	-3.3987	0.0030	-4.0372	0.00070	-3.9784	0.0008
Single Rank	0.6545	***0.5206***	-5.6385	1.95E-05	-0.9103	*0.3740*

ery node of the EdgeGraph represents bi-gram. In this section, we analyse the results on WCN and EdgeGraph text representation for different characteristics of datasets.

4.5.1 Polish and Portuguese dataset

The random walk based GKET for Portuguese and Polish language over EdgeGraph shows major improvements with *TextRank*, *SingleRank and PositionRank*. In future, the robustness and the scalability of EdgeGraph over non-English datasets can be tested for long textual documents, different languages and for different domains.

4.5.2 Medium-sized textual documents

The English language medium-sized dataset outperforms all other datasets with EdgeGraph. It is interesting to note that though there is improvement for medium-sized textual documents, the resulting values for English and Portuguese dataset are promising but not suitable for Polish dataset.

4.5.3 Long-text documents

The long-text datasets: Wiki20, show no or slight improvement with F-measure but significant improvement with recall. The nodes represent bigrams in the EdgeGraph due to which more number of words are obtained. Hence, recall is improved more than the precision. The EdgeGraph representation gives better results over medium-sized text (containing less than 500 words) as compared to long text (containing more than 4000 words) irrespective of the language.

4.5.4 Varying number of documents

The number of documents in different dataset varies from 20 to 500 which may affect the resulting values. More the number of documents, the stronger the results. We found that the datasets with large number of documents such as 500N-KPC and 110-PT-BN-KP shows consistency over improvements for all the random walk based GKET and gives improved F-measure for all the random walk based GKET.

4.6 Statistical Significance

The results obtained by exploiting random walk based GKET over WCN and EdgeGraph are not directly comparable. We further investigate the improvements to test and validate the robustness and significance of the results. We use the *Student's t-test* with 5% of significance level. The statistical significance is evaluated over the resulting values of k varying from 1 to 20 as shown in Table 6. The null-hypothesis in $t - test$ is that the two series of resulting values are significantly different if the $p - value < 0.5$. We use the following symbolic representation for four categories of statistical analysis:

1. *EdgeGraph significantly outperforms WCN*: We represent *Bold* $p - value$ if $t - test$ is positive and the $p - value < 0.05$.

2. *EdgeGraph is better than WCN, but not statistically significant*: We represent *bold + italics* $p - value$ if $t - test$ are positive and the $p - value 0.05$.

3. *WCN is better than EdgeGraph, but not statistically significant*: We represent *italics* $p - value$ if $t - test$ are negative and the $p - value 0.05$.

4. *WCN significantly outperforms EdgeGraph*: We represent normally formatted $p - value$ if $t - test$ are negative and the $p - value < 0.05$

We investigate the improvements with similar and comparative performance of EdgeGraph over the WCN. In this context, we consider first three cases to signify non-deteriorating measure. We found that the Recall and F-measure shows good performance with EdgeGraph in 83.33% and 66.66% of the total number of cases. On the basis of individual performance, the SingleRank outperforms all other random walk based GKET.

5 Conclusion

Here in this work, we propose an EdgeGraph representation for information retrieval tasks. The experimental results over four publicly available datasets shows that keyphrase extraction is significantly improved with EdgeGraph representation leveraging on bi-grams. The recall and F- measure improves upto 27% and 18%, respectively, for the datasets with medium-sized English texts. Applicability of EdgeGraph on more than one languages (English and Portuguese) suggests its language-independencex. In future, EdgeGraph can be used for extractive text summarization, language generation and cross-lingual analysis and other information retrieval tasks. In addition to this, the massive online data can be handled using dynamics of EdgeGraph evolved from dynamically streaming data without using any pre-trained or supervised models.

References

Willyan D Abilhoa and Leandro N De Castro. 2014. A keyword extraction method from twitter messages represented as graphs. *Applied Mathematics and Computation*, 240:308–325.

Mubashar Nazar Awan and Mirza Omer Beg. 2021. Top-rank: a topicalpostionrank for extraction and classification of keyphrases in text. *Computer Speech & Language*, 65:101116.

Slobodan Beliga, Ana Meštrović, and Sanda Martinčić-Ipšić. 2016. Selectivity-based keyword extraction method. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 12(3):1–26.

Abdelghani Bellaachia and Mohammed Al-Dhelaan. 2012. Ne-rank: A novel graph-based keyphrase extraction in twitter. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 372–379. IEEE.

Saroj Kr Biswas, Monali Bordoloi, and Jacob Shreya. 2018. A graph based keyword extraction model using collective node weight. *Expert Systems with Applications*, 97:51–59.

Florian Boudin. 2018. Unsupervised keyphrase extraction with multipartite graphs. *arXiv preprint arXiv:1803.08721*.

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. Topicrank: Graph-based topic ranking for keyphrase extraction. In *International joint conference on natural language processing (IJCNLP)*, pages 543–551.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. 2020. Yake! keyword extraction from single documents using multiple local features. *Information Sciences*, 509:257–289.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Célia Nunes, and Adam Jatowt. 2018. A text feature based automatic keyword extraction method for single documents. In *European conference on information retrieval*, pages 684–691. Springer.

Monojit Choudhury, Diptesh Chatterjee, and Animesh Mukherjee. 2010. Global topology of word co-occurrence networks: Beyond the two-regime power-law. In *Coling 2010: Posters*, pages 162–170.

Swagata Duari and Vasudha Bhatnagar. 2019. scake: semantic connectivity aware keyword extraction. *Information Sciences*, 477:100–117.

Corina Florescu and Cornelia Caragea. 2017. Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1115.

Muskan Garg. 2021. A survey on different dimensions for graphical keyword extraction techniques. *Artificial Intelligence Review*, pages 1–40.

Muskan Garg and Mukesh Kumar. 2018a. Identifying influential segments from word co-occurrence networks using ahp. *Cognitive Systems Research*, 47:28–41.

Muskan Garg and Mukesh Kumar. 2018b. The structure of word co-occurrence network for microblogs. *Physica A: Statistical Mechanics and its Applications*, 512:698–720.

Muskan Garg and Mukesh Kumar. 2020. Finding summaries to obtain event phrases from streaming microblogs using word co-occurrence network. In *2020 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pages 200–206. IEEE.

Lise Getoor and Christopher P Diehl. 2005. Link mining: a survey. *Acm Sigkdd Explorations Newsletter*, 7(2):3–12.

Lu Huang, Xiang Chen, Xingxing Ni, Jiarun Liu, Xiaoli Cao, and Changtian Wang. 2021. Tracking the dynamics of co-word networks for emerging topic identification. *Technological Forecasting and Social Change*, 170:120944.

Marie Katsurai. 2017. Bursty research topic detection from scholarly data using dynamic co-word networks: A preliminary investigation. In *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, pages 115–119. IEEE.

Ashkan Kazemi, Verónica Pérez-Rosas, and Rada Mihalcea. 2020. Biased textrank: Unsupervised graph-based content extraction. *arXiv preprint arXiv:2011.01026*.

Wei Liang. 2017. Spectra of english evolving word co-occurrence networks. *Physica A: Statistical Mechanics and its Applications*, 468:802–808.

Wei Liang, Yuming Shi, K Tse Chi, Jing Liu, Yanli Wang, and Xunqiang Cui. 2009. Comparison of co-occurrence networks of the chinese and english languages. *Physica A: Statistical Mechanics and its Applications*, 388(23):4901–4909.

HaiTao Liu and Jin Cong. 2013. Language clustering with word co-occurrence networks based on parallel texts. *Chinese Science Bulletin*, 58(10):1139–1144.

Yonghe Lu, Jiayi Luo, Ying Xiao, and Hou Zhu. 2021. Text representation model of scientific papers based on fusing multi-viewpoint information and its quality assessment. *Scientometrics*, pages 1–27.

Luis Marujo, Márcio Viveiros, and João Paulo da Silva Neto. 2013. Keyphrase cloud generation of broadcast news. *arXiv preprint arXiv:1306.4606*.

Olena Medelyan, Ian H Witten, and David Milne. 2008. Topic indexing with wikipedia. In *Proceedings of the AAAI WikiAI workshop*, volume 1, pages 19–24.

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.

Ahmed Hamza Osman and Omar Mohammed Barukub. 2020. Graph-based text representation and matching: A review of the state of the art and future challenges. *IEEE Access*, 8:87562–87583.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Eirini Papagiannopoulou, Grigorios Tsoumakas, and Apostolos Papadopoulos. 2021. Keyword extraction using unsupervised learning on the document's adjacency matrix. In *Proceedings of the Fifteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-15)*, pages 94–105.

Laura VC Quispe, Jorge AV Tohalino, and Diego R Amancio. 2021. Using virtual edges to improve the discriminability of co-occurrence text networks. *Physica A: Statistical Mechanics and its Applications*, 562:125344.

Zafar Saeed, Rabeeh Ayaz Abbasi, Muhammad Imran Razzak, and Guandong Xu. 2019. Event detection in twitter stream using weighted dynamic heartbeat graph approach [application notes]. *IEEE Computational Intelligence Magazine*, 14(3):29–38.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077.

Antoine Tixier, Fragkiskos Malliaros, and Michalis Vazirgiannis. 2016. A graph degeneracy-based approach to keyword extraction. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1860–1870.

Asahi Ushio, Federico Liberatore, and Jose Camacho-Collados. 2021. Back to the basics: A quantitative analysis of statistical and graph-based term weighting schemes for keyword extraction. *arXiv preprint arXiv:2104.08028*.

Javier Vera and Wenceslao Palma. 2021. The community structure of word co-occurrence networks: Experiments with languages from the americas. *EPL (Europhysics Letters)*, 134(5):58002.

Xiaojun Wan and Jianguo Xiao. 2008a. Collabrank: towards a collaborative approach to single-document keyphrase extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 969–976.

Xiaojun Wan and Jianguo Xiao. 2008b. Single document keyphrase extraction using neighborhood knowledge. In *AAAI*, volume 8, pages 855–860.

Mingxi Zhang, Xuemin Li, Shuibo Yue, and Liuqian Yang. 2020. An empirical study of textrank for keyword extraction. *IEEE Access*, 8:178849–178858.

Massively Multilingual Language Models for Cross Lingual Fact Extraction from Low Resource Indian Languages

Bhavyajeet Singh[*], **Pavan Kandru**[*], **Anubhav Sharma** and **Vasudeva Varma**
Information Retrieval and Extraction Lab, IIIT Hyderabad, India
{bhavyajeet.singh, siri.venkata, anubhav.sharma}@research.iiit.ac.in
vv@iiit.ac.in

Abstract

Massive knowledge graphs like Wikidata attempt to capture world knowledge about multiple entities. Recent approaches concentrate on automatically enriching these KGs from text. However a lot of information present in the form of natural text in low resource languages is often missed out. Cross Lingual Information Extraction aims at extracting factual information in the form of English triples from low resource Indian Language text. Despite its massive potential, progress made on this task is lagging when compared to Monolingual Information Extraction. In this paper, we propose the task of Cross Lingual Fact Extraction(CLFE) from text and devise an end-to-end generative approach for the same which achieves an overall F1 score of 77.46.

1 Introduction

Knowledge graphs are large structured sources of information about the world. Recently, a lot of attention is being put in finding ways to automatically build or enrich extensive knowledge graphs (KGs) (Gupte et al., 2021), (Zou, 2020). Wikidata (Vrandečić and Krötzsch, 2014) is one of the largest publicly available knowledge graphs which has over 99 million entities. Knowledge graphs such as Wikidata have been extensively used for multiple applications like text generation (Koncel-Kedziorski et al., 2019), question answering (Srivastava et al., 2021), (Li et al., 2022) etc.

A knowledge graph is composed of multiple facts linked together. A fact is often represented as a triplet which consists of two entities and a semantic relation between them. This information can be encoded as a triple $< h, r, t >$ where h is the subject entity, r is the relation and t represents the tail entity.

Fact extraction refers to the task of extracting structured factual information from natural language text (Charu C. Aggarwal, 2012). Previously there has been extensive work regarding the task of monolingual fact extraction, especially in English (Zeng et al., 2019) (Li et al., 2021), however not much attention has been given to the task of cross-lingual fact extraction.

In this paper we propose an important task of multi-lingual and cross-lingual fact to text extraction(CLFE) for 7 Low Resource(LR) Indian Languages and English. The task aims at directly extracting English triples from 8 different languages. We also propose strong baselines and approaches for this task which produce results comparable to existing mono-lingual state-of-the-art fact extraction pipelines and significantly better than other previous cross lingual attempts at fact extraction (Zhang et al., 2017a). Our work enables utilisation of factual knowledge present in Indic texts in order to increase the coverage of existing knowledge graphs. This would further help in multiple downstream tasks like fact verification, text generation etc. To the best of our knowledge, this is the first attempt at multilingual and cross-lingual fact extraction from LR Indian Languages. Figure 1 shows multiple examples of the input and output for CLFE task.

Overall, we make the following contributions. (1) Propose the problem of cross-lingual and multilingual fact extraction for LR Indian languages. (2) An end-to-end generative approach for extracting subject centric factual information from LR Indian language text, which shows significant improvements over classification based pipelines. (3) We train multiple multi-lingual CLFE models which lead to an overall F1 score of 77.46 .

2 Related work

Extracting structured information from free form text is a problem well worked upon. T-REx (Elsahar et al., 2018) uses entity linking, co-reference resolution and string match based link-

[*] Equal contribution

code available at https://github.com/bhavyajeet/CLFE

Multilingual Texts	Head Entity	Extracted Facts
Elon Reeve Musk (born 28 June 1971) is a South African-Canadian-America n veteran businessman, investor, engineer, and inventor.	Elon Musk	< nationality, South_Africa> < nationality, Canda> < nationality, USA> < date_of_birth, 28_June_1971> < occupation, engineer> < ocuupation, entrepreneur> < occupation, inventor> < occupation, investor>
एलन मस्क (जन्म 28 जून 1971) एक दक्षिण अफ्रीकी-कनाडाई-अमेरिकी दिग्गज व्यापारी, निवेशक, इंजीनियर, और आविष्कारक हैं।	Elon Musk	< nationality, South_Africa> < nationality, Canda> < nationality, USA> < date_of_birth, 28_June_1971> < occupation, engineer> < ocuupation, entrepreneur> < occupation, inventor> < occupation, investor>
এলন মাস্ক (জন্ম 28 জুনজু 1971) দক্ষিণ আফ্রিকা-কানাডিয়ান-আমেরিকান প্রবীণ ব্যবসায়ী, বিনিয়োগকারী, প্রকৌশলী এবং উদ্ভাবক।	Elon Musk	< nationality, South_Africa> < nationality, Canda> < nationality, USA> < date_of_birth, 28_June_1971> < occupation, engineer> < ocuupation, entrepreneur> < occupation, inventor> < occupation, investor>
એલોન મસ્ક (જન્મ 28 જૂન 1971) એ દક્ષિણ આફ્રિકા-કેનેડિયન□અમેરિકન પીte ઉદ્યોગપતિ, રોકાણકાર, ઇજનેર અને શોધક છે.	Elon Musk	< nationality, South_Africa> < nationality, Canda> < nationality, USA> < date_of_birth, 28_June_1971> < occupation, engineer> < ocuupation, entrepreneur> < occupation, inventor> < occupation, investor>

Figure 1: Example Inputs and outputs of CLFE task. Text from any language along with entity of interest(head entity) is provided as input to extract English Facts(relation and tail entity pairs). The same sentence may or may not be present in all languages.

ing pipelines to perform fact linking between DBPedia (Lehmann et al., 2015) abstracts and Wikidata (Vrandečić and Krötzsch, 2014) triples. REFCOG (Kolluru et al., 2021) works in a cross lingual space to link the facts and outperforms the existing pipeline based approaches like (Elsahar et al., 2018). But these approaches are limited in their application since they perform fact linking and need a fact set as input.

OpenIE(Angeli et al., 2015) tackles this issue by leveraging linguistic structure for open domain information extraction. While the predecessor open domain IE systems like Ollie(Mausam et al., 2012) use large set of patterns with broad coverage to extract facts, OpenIE uses a small set of patterns which works well on canonically structured sentences. However, these open domain information extractors produce facts that have long and over-specific relations which can not be used to construct KGs.

(Zhong and Chen, 2020); (Sui et al., 2020a) approach the information extraction problem by jointly extracting entities and their relation from input text using neural models without referring to any repository of facts. Although these works produce systems which can extract open information from text in the WebNLG dataset (Gardent et al., 2017), they are monolingual and are limited to knowledge extraction in a single language. Various existing well performing relation extraction models like (Yan et al., 2021), (Sui et al., 2020b) rely partially on exact match of entities in the source text, which makes it harder to adapt them for the CLFE task.

Cross Lingual fact extraction i.e extracting facts from source text of different languages didn't receive as much attention as Monolingual Fact extraction did. Although (Zhang et al., 2017b) worked on this task, with just a single language, the highest reported f1 is 33.67. Moreover, Fact extraction from low resource languages like Indic Languages hasn't been attempted. In this work, we attempt to reduce these gaps in Information extraction by proposing systems for Cross Lingual Subject Centric Fact Extraction in low resource Indic Languages.

3 Dataset

For the task of CLFE we leverage the XAlign dataset (Abhishek et al., 2022). The dataset contains 0.45M pairs across 8 languages, of which 5402 pairs have been manually annotated. The manually annotated part of the dataset was used as the golden test set. The sentences in XAlign come from the Wikipedia articles, about entities belonging to the human class, written in Indian languages.

The extensively cross lingual and multi lingual nature of the XAlign dataset is ideal for the proposed task. Though originally designed for the task of cross lingual data to text generation, the XAlign dataset can be leveraged for CFLE as well. However the dataset poses certain challenges. If we were to consider each relation as a class (for classification based approaches), the dataset is highly imbalanced. Out of approximately 367 unique relations(classes), the most frequent class alone has a frequency of 27 % and top 20 classes contribute to 90% of the data. The data contains an average of 2.02 facts aligned per sentence.

Along with this, another challenging aspect of the dataset is that it is partially aligned. While the sentences in the test set have complete coverage in the aligned facts, the entire information present in the sentences from the train set is not covered by the aligned facts. This attribute of the dataset,

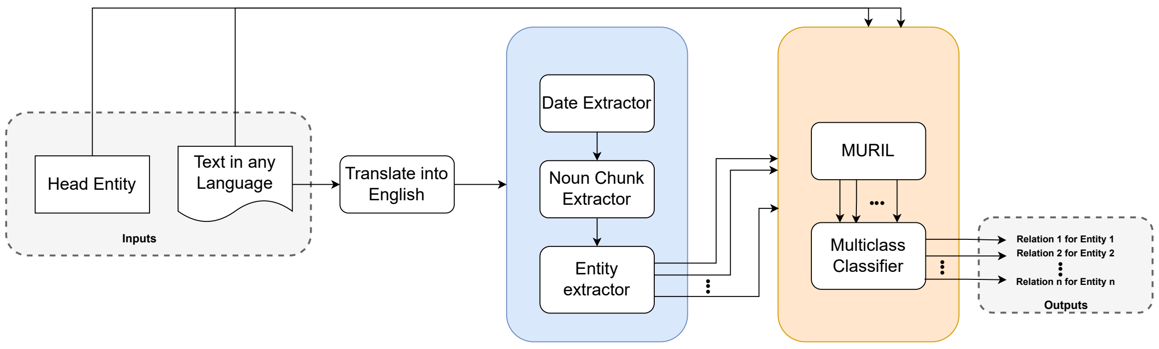


Figure 2: Pipeline Architecture for CLFE

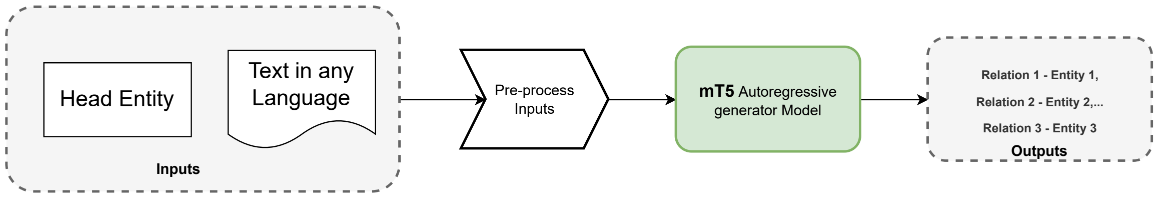


Figure 3: End to end architecture for CLFE

can potentially penalise the model even for the generation of correct facts during the training time. Thus impacting recall scores during the test time. More details in Appendix section A.2

4 Methodology

We propose two approaches for the CLFE task. The first approach is a classification based approach which extracts tails first and then predicts the relation. Second approach is a generative one that does both of these task in one shot.

4.1 Tail Extraction and Relation Classification(TERC)

The TERC pipeline (Figure 2) consists of two steps. The first step is to extract tails of facts from the source language text. To do this we use IndicTrans (Ramesh et al., 2021) translation and convert input text to English language. After this we extract any dates present in the text and normalize them in to the same format. We also replace them in the original text with a dummy token to preclude dates from participating in other entities. Since every tail entity can only be a noun or proper noun, we use spaCy (Honnibal and Montani, 2018) noun chunk extractor to extract all the noun chunks from which tail entities are selected as follows.

- Entities that match with head are removed. Since we are only interested about tails at this stage of the pipeline we remove any entities that have high lexical overlap with head.
- All noun chunks with pronoun roots are removed to filter pronouns. Tails present in the data are never pronouns so we prune out all the recognized phrases which have pronoun heads.
- Continuous spans of tokens with ADJ and PROPN PoS tags are selected as individual entities. Tails are multi word entities and may contain adjectives within their span, so we use PoS tags to get maximal spans for every detected proper nouns.
- Root of the noun chunk is selected as a separate entity if its PoS tag is NOUN.

Next step is to predict a relation for each of these tails. To do this we use pretrained MuRIL (Khanuja et al., 2021) to generate a joint representation of head entity, tail entity and source language input text. This representation is fed as input to a classifier which predicts the relation between the head and the tail entities in the input. The classifier is trained on the training set to predict the relation, given a sentence and a <head, tail> pair, by considering the tails from ground truth as input. In order to tackle the class imbalance, we use **inverse log of class distribution** as weights in loss-function which performs better than standard inverse class distribution as well as unweighted loss.

While evaluating the performance of the pipeline architecture, tails extracted from translated input text, are aligned with ground truth tails. The details of this alignment are described in A.1 of the Ap-

	te	bn	ta	gu	mr	en	hi	kn	All languages		
	F1	F1	F1	F1	F1	F1	F1	F1	P	R	F1
Classification with GT Tails	69.19	67.50	89.44	85.74	51.38	72.87	87.10	79.74	79.04	77.93	75.37
TERC	43.66	41.96	52.19	40.30	44.59	50.80	50.46	42.57	40.45	53.71	46.15
E2E Cross-lingual Generative Model	71.82	75.56	82.82	**72.36**	**77.79**	76.28	**86.62**	68.04	74.09	**81.15**	**77.46**
E2E generation w script unification	**72.51**	75.38	**85.21**	72.04	77.19	74.56	83.44	**70.46**	78.49	76.15	77.29
Bilingual Models	70.94	**78.01**	83.71	67.84	71.91	**76.64**	86.49	63.19	**79.79**	71.63	75.49

Table 1: Precision, recall and F1 scores of various methods applied on all languages in the Test set. Note that "Classification with GT Tails" uses tails from ground truth as input for the Relation Prediction model and hence does not represent a complete pipeline

pendix. Predictions are made for the aligned tails and evaluation metrics are calculated on the same.

4.2 End to End Generative extraction

We also propose an end to end approach (Figure 3) to the fact extraction problem which can jointly extract tails as well as their relations with the head entity. Previous work in the domain of monolingual fact extraction has shown that a model which jointly performs the tail and relation extraction is more likely to perform better than a disjoint approach (Li et al., 2021). Advantage of this approach over the pipeline approach mentioned above is that there is a two way interaction between tail extraction and relation prediction which improves performance of both the tasks as they are not independent of each other.

We pose this problem as a text-to-text task and use the mT5 (Xue et al., 2020) auto-regressive seq-2-seq model to generate relations and tails, when head entity and input text are given as inputs. We use cross entropy loss to train this model. Using a generative model allows for a more generalizable and open information extraction i.e set of relations and tails are not restricted.

We experiment with 3 variations of this pipeline. In all these variations, the facts are linearised as the target text by concatenating the head and tail joined by special tokens. Thus for a given sentence S, if the corresponding i facts are $[h, r_1, t_1]$, $[h, r_2, t_2]$....$[h, r_i, t_i]$, the target text would be $< R > r_1 < T > t_1 < R > r_2 < T > t_2 .... < R > r_i < T > t_i$.

The first variation is fine-tuning the pretrained mt5 model for the fact extraction task over all languages. For the second experiment, we use script unification where we transliterate the input text of all languages except English to the Devanagari script. The idea is that the unified script input helps the model's training due to a high overlap in the vocabulary accross multiple Indian languages. In our third variation, we train multiple bi-lingual fact extraction models, one for each language. The implementation details of regarding these models and TERC(4.1) are in the Appendix A.3.

5 Results and Discussion

Table 1 summarizes the results of the multiple fact extraction approaches mentioned in section 4.

It can be observed that the open ended approach performs the best in terms of F1 score while also providing complete flexibility regarding the possible entities and relations. Another observation is that the strategy where we train separate bilingual models, works better than the combined model for just two languages, English and Bengali. This is explained by the fact that these are the two most frequent languages for our dataset, which together constitute 54.44 % of our training data. Thus, multilingual training proves to be useful over all, because of the shared learning across Indian languages. We also observe that script unification (transliterating input scripts to Devanagari), specifically benefits all the Dravidian languages (te, ta, kn) of our dataset.

It should be noted that the actual performance of the model might be better than what the numbers show. The reason for this is that currently we adhere to strict evaluation schemes where a word match between the predicted and the actual tail is necessary in order to determine the prediction as correct. However, this misses out on cases where the predicted and the ground truth tails are completely synonymous. An example of this is the case where the model predicts the occupation as 'writer', whereas the GT label has it as 'author'.

6 Conclusion and Future work

In this work, we introduce the task of multilingual and cross-lingual fact extraction over English and seven other LR Indic languages. We conclude that though script-unification helps certain languages, a single multilingual end-to-end generative pipeline performs better with overall F1 score of 77.46. This work paves the path for upcoming research in methods of extracting knowledge from LR Indic language text. In future, we plan to explore approaches that make specific effort to tackle the partially aligned nature of the dataset in order to achieve further improvements.

References

Tushar Abhishek, Shivprasad Sagare, Bhavyajeet Singh, Anubhav Sharma, Manish Gupta, and Vasudeva Varma. 2022. Xalign: Cross-lingual fact-to-text alignment and generation for low-resource languages. In *Companion Proceedings of the Web Conference 2022*, WWW '22, page 171–175, New York, NY, USA. Association for Computing Machinery.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China. Association for Computational Linguistics.

ChengXiang Zhai Charu C. Aggarwal. 2012. Mining text data.

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)*, Miyazaki, Japan. European Language Resources Association (ELRA).

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.

Athang Gupte, Saumitra Sapre, and Sheetal Sonawane. 2021. Knowledge graph generation from text using neural machine translation techniques. In *2021 International Conference on Communication information and Computing Technology (ICCICT)*, pages 1–8.

Matthew Honnibal and Ines Montani. 2018. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. *To appear*.

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 Talukdar. 2021. Muril: Multilingual representations for indian languages.

Keshav Kolluru, Martin Rezk, Pat Verga, William W. Cohen, and Partha Talukdar. 2021. Multilingual fact linking.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text generation from knowledge graphs with graph transformers.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Ren Li, Dong Li, Jianxi Yang, Fangyue Xiang, Hao Ren, Shixin Jiang, and Luyi Zhang. 2021. Joint extraction of entities and relations via an entity correlated attention neural model. *Information Sciences*, 581:179–193.

Sirui Li, Kok Kai Wong, Dengya Zhu, and Chun Che Fung. 2022. Improving question answering over knowledge graphs using graph summarization.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534, Jeju Island, Korea. Association for Computational Linguistics.

Gowtham Ramesh, Sumanth Doddapaneni, Aravinth Bheemaraj, Mayank Jobanputra, Raghavan AK, Ajitesh Sharma, Sujit Sahoo, Harshita Diddee, Mahalakshmi J, Divyanshu Kakwani, Navneet Kumar, Aswin Pradeep, Srihari Nagaraj, 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.

Saurabh Srivastava, Mayur Patidar, Sudip Chowdhury, Puneet Agarwal, Indrajit Bhattacharya, and Gautam Shroff. 2021. Complex question answering on knowledge graphs using machine translation and multi-task learning. In *Proceedings of the 16th Conference of* *the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3428–3439, Online. Association for Computational Linguistics.

Dianbo Sui, Yubo Chen, Kang Liu, Jun Zhao, Xiangrong Zeng, and Shengping Liu. 2020a. Joint entity and relation extraction with set prediction networks.

Dianbo Sui, Yubo Chen, Kang Liu, Jun Zhao, Xiangrong Zeng, and Shengping Liu. 2020b. Joint entity and relation extraction with set prediction networks.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

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.

Zhiheng Yan, Chong Zhang, Jinlan Fu, Qi Zhang, and Zhongyu Wei. 2021. A partition filter network for joint entity and relation extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 185–197, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Daojian Zeng, Ranran Haoran Zhang, and Qianying Liu. 2019. Copymtl: Copy mechanism for joint extraction of entities and relations with multi-task learning.

Sheng Zhang, Kevin Duh, and Benjamin Van Durme. 2017a. MT/IE: Cross-lingual open information extraction with neural sequence-to-sequence models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 64–70, Valencia, Spain. Association for Computational Linguistics.

Sheng Zhang, Kevin Duh, and Benjamin Van Durme. 2017b. MT/IE: Cross-lingual open information extraction with neural sequence-to-sequence models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 64–70, Valencia, Spain. Association for Computational Linguistics.

Zexuan Zhong and Danqi Chen. 2020. A frustratingly easy approach for entity and relation extraction.

Xiaohan Zou. 2020. A survey on application of knowledge graph. *Journal of Physics: Conference Series*, 1487(1):012016.

A Example Appendix

A.1 Tail Entity Alignment

Entities extracted from the translated texts are aligned with the gold truth tail entities in order to measure performance on test set. By alignment we mean that we assign one ground truth tail entity to each extracted entity without repetition. Some extracted entities which do not have any overlap with ground truth are ignored. Some ground truth entities might not be assigned any of the extracted entities leading to lower recall. Assignment is done based on a similarity score and a threshold. Similarity score between two entities is calculated as the sum of cosine similarities of GloVe vectors and intersection over union of terms. With a threshold of 0.7 we achieved a precision of 0.54 and a recall of 0.77.

A.2 Additional Dataset Statistics

Figure 4 shows the distribution of Top 30 most frequent relations in the dataset. Figure 5 depicts the share of each of the languages in the dataset. As it can be seen, the dataset is highly imbalanced both in terms of relations and languages.

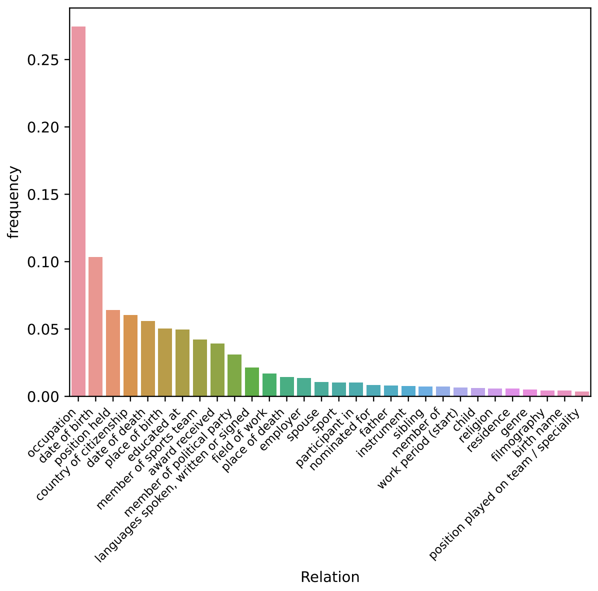


Figure 4: Distribution of Top 30 most frequent relations in the dataset

A.3 Implementation Details

Both Two-Phase and E2E generative architectures are trained on NVIDIA GeForce RTX 2080 Ti graphic cards. For the Two-Phase approach, the

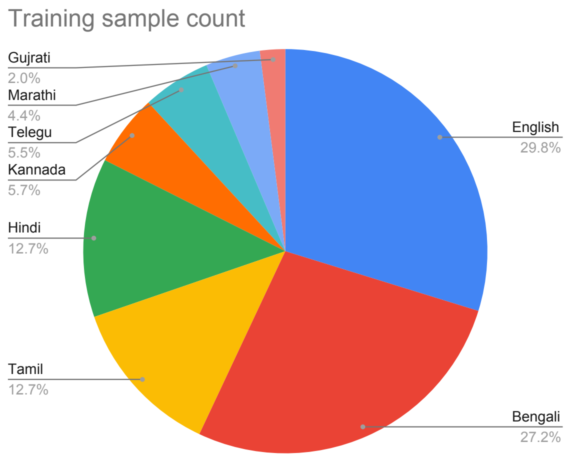


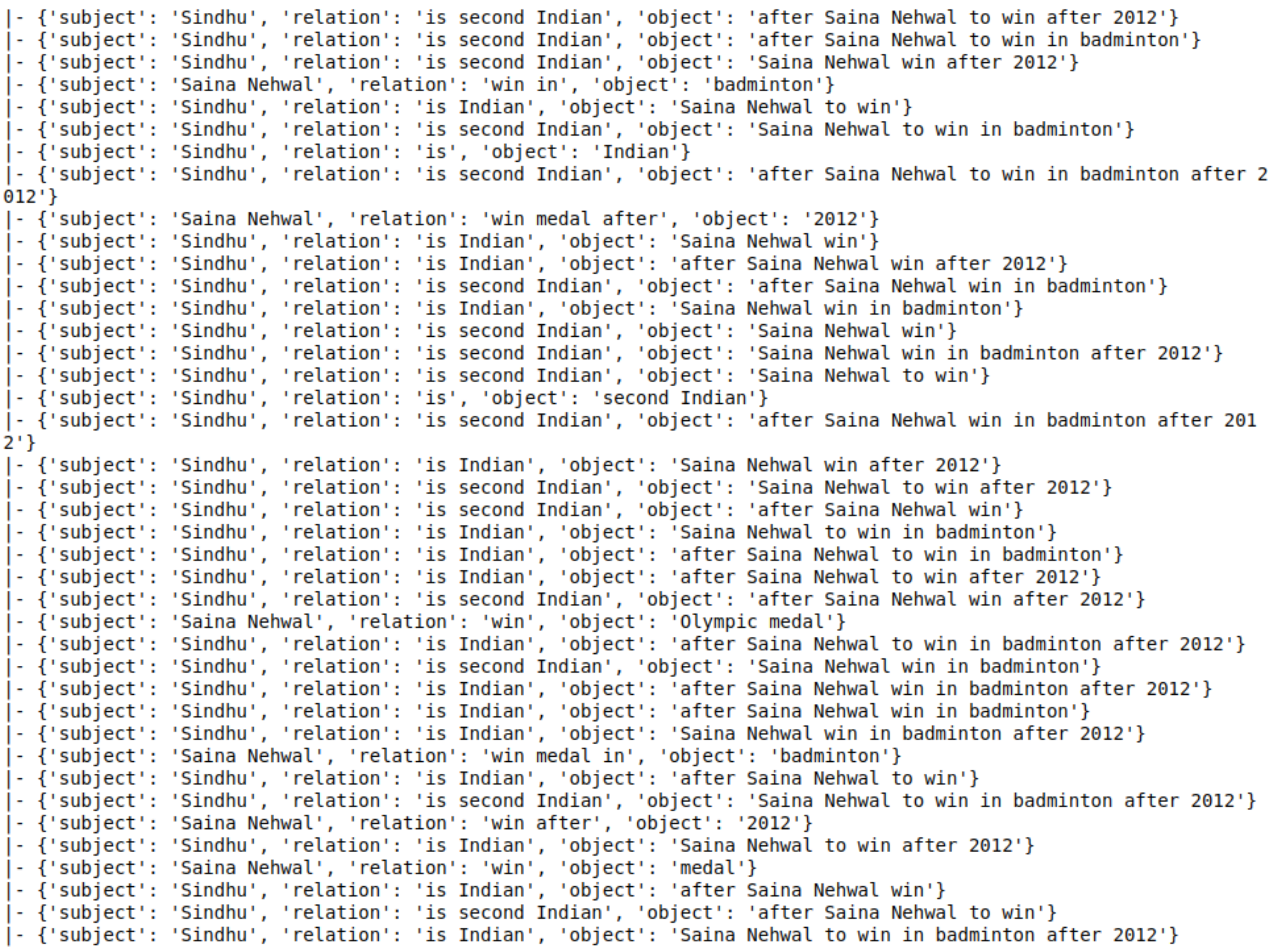
Figure 5: Distribution of the 8 languages in the training set

only block that needed training is relation prediction. MURIL encoder model from google which has 12 encoding layers and output dimension 768 is the base of the classifier. 12th layer of MURIL along with the layers in the feed forward network are unfrozen during the training phase. Adam optimizer is used with initial learning rate of 1e-4 and step scheduling with step size 2 and gamma 0.3. Batches of 16 facts are trained to optimize Cross Entropy Loss. Inverse log frequency of classes is used as weights for cross entropy loss to counteract the imbalance in the dataset. Training relation prediction 5 hours on 1 GPU card.

For the Generative approach, we used the pretrain mT5 model and finetune it for 5 epochs for all experiments. The learning rate is 0.001 with a weight decay of 0.01. The dropout rate is set to 0.1 in order to prevent over fitting on the training data. We use the Adafactor optimizer to optimise the Cross Entropy Loss during generation.

A.4 Analysis of OpenIE extraction

We tried openIE extractor from stanford to extract from english translated versions of texts from other languages. Even after discounting translation losses facts extracted from openIE were not useful because of overly specific relations and entities. Figure 6 is an example for the source sentence "Sindhu is the second Indian after Saina Nehwal to win in badminton after 2012"



```
|- {'subject': 'Sindhu', 'relation': 'is second Indian', 'object': 'after Saina Nehwal to win after 2012'}
|- {'subject': 'Sindhu', 'relation': 'is second Indian', 'object': 'after Saina Nehwal to win in badminton'}
|- {'subject': 'Sindhu', 'relation': 'is second Indian', 'object': 'Saina Nehwal win after 2012'}
|- {'subject': 'Saina Nehwal', 'relation': 'win in', 'object': 'badminton'}
|- {'subject': 'Sindhu', 'relation': 'is Indian', 'object': 'Saina Nehwal to win'}
|- {'subject': 'Sindhu', 'relation': 'is second Indian', 'object': 'Saina Nehwal to win in badminton'}
|- {'subject': 'Sindhu', 'relation': 'is', 'object': 'Indian'}
|- {'subject': 'Sindhu', 'relation': 'is second Indian', 'object': 'after Saina Nehwal to win in badminton after 2
012'}
|- {'subject': 'Saina Nehwal', 'relation': 'win medal after', 'object': '2012'}
|- {'subject': 'Sindhu', 'relation': 'is Indian', 'object': 'Saina Nehwal win'}
|- {'subject': 'Sindhu', 'relation': 'is Indian', 'object': 'after Saina Nehwal win after 2012'}
|- {'subject': 'Sindhu', 'relation': 'is second Indian', 'object': 'after Saina Nehwal win in badminton'}
|- {'subject': 'Sindhu', 'relation': 'is Indian', 'object': 'Saina Nehwal win in badminton'}
|- {'subject': 'Sindhu', 'relation': 'is second Indian', 'object': 'Saina Nehwal win'}
|- {'subject': 'Sindhu', 'relation': 'is second Indian', 'object': 'Saina Nehwal win in badminton after 2012'}
|- {'subject': 'Sindhu', 'relation': 'is second Indian', 'object': 'Saina Nehwal to win'}
|- {'subject': 'Sindhu', 'relation': 'is', 'object': 'second Indian'}
|- {'subject': 'Sindhu', 'relation': 'is second Indian', 'object': 'after Saina Nehwal win in badminton after 201
2'}
|- {'subject': 'Sindhu', 'relation': 'is Indian', 'object': 'Saina Nehwal win after 2012'}
|- {'subject': 'Sindhu', 'relation': 'is second Indian', 'object': 'Saina Nehwal to win after 2012'}
|- {'subject': 'Sindhu', 'relation': 'is second Indian', 'object': 'after Saina Nehwal win'}
|- {'subject': 'Sindhu', 'relation': 'is Indian', 'object': 'Saina Nehwal to win in badminton'}
|- {'subject': 'Sindhu', 'relation': 'is Indian', 'object': 'after Saina Nehwal to win in badminton'}
|- {'subject': 'Sindhu', 'relation': 'is Indian', 'object': 'after Saina Nehwal to win after 2012'}
|- {'subject': 'Sindhu', 'relation': 'is second Indian', 'object': 'after Saina Nehwal win after 2012'}
|- {'subject': 'Saina Nehwal', 'relation': 'win', 'object': 'Olympic medal'}
|- {'subject': 'Sindhu', 'relation': 'is Indian', 'object': 'after Saina Nehwal to win in badminton after 2012'}
|- {'subject': 'Sindhu', 'relation': 'is second Indian', 'object': 'Saina Nehwal win in badminton'}
|- {'subject': 'Sindhu', 'relation': 'is Indian', 'object': 'after Saina Nehwal win in badminton after 2012'}
|- {'subject': 'Sindhu', 'relation': 'is Indian', 'object': 'after Saina Nehwal win in badminton'}
|- {'subject': 'Sindhu', 'relation': 'is Indian', 'object': 'Saina Nehwal win in badminton after 2012'}
|- {'subject': 'Saina Nehwal', 'relation': 'win medal in', 'object': 'badminton'}
|- {'subject': 'Sindhu', 'relation': 'is Indian', 'object': 'after Saina Nehwal to win'}
|- {'subject': 'Sindhu', 'relation': 'is second Indian', 'object': 'Saina Nehwal to win in badminton after 2012'}
|- {'subject': 'Saina Nehwal', 'relation': 'win after', 'object': '2012'}
|- {'subject': 'Sindhu', 'relation': 'is Indian', 'object': 'Saina Nehwal to win after 2012'}
|- {'subject': 'Saina Nehwal', 'relation': 'win', 'object': 'medal'}
|- {'subject': 'Sindhu', 'relation': 'is Indian', 'object': 'after Saina Nehwal win'}
|- {'subject': 'Sindhu', 'relation': 'is second Indian', 'object': 'after Saina Nehwal to win'}
|- {'subject': 'Sindhu', 'relation': 'is Indian', 'object': 'Saina Nehwal to win in badminton after 2012'}
```

Figure 6: Examples of output from OpenIE

Analysing Syntactic and Semantic Features in Pre-trained Language Models in a Fully Unsupervised Setting

Necva Bölücü
Computer Engineering
Hacettepe University
Adana Alparslan Turkes
Science and Technology University
Adana, Turkey
nbolucu@atu.edu.tr

Burcu Can
RGCL, University of Wolverhampton
Wolverhampton, UK
b.can@wlv.ac.uk

Abstract

Transformer-based pre-trained language models (PLMs) have been used in all NLP tasks and resulted in a great success. This has led to the question of whether we can transfer this knowledge to syntactic or semantic parsing in a completely unsupervised setting. In this study, we leverage PLMs as a source of external knowledge to perform a fully unsupervised parser model for semantic, constituency and dependency parsing. We analyse the results for English, German, French, and Turkish to understand the impact of the PLMs on different languages for syntactic and semantic parsing. We visualize the attention layers and heads in PLMs for parsing to understand the information that can be learned throughout the layers and the attention heads in the PLMs both for different levels of parsing tasks. The results obtained from dependency, constituency, and semantic parsing are similar to each other, and the middle layers and the ones closer to the final layers have more syntactic and semantic information.

1 Introduction

Transformer-based pre-trained language models (PLMs) such as BERT (Devlin et al., 2019), GPT-2 (Radford et al., 2019), XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019b), DistilBERT (Sanh et al., 2019) have shown state-of-art performance in many down-stream NLP tasks. The performance of such large PLMs has also begged the question of what type of information that these models can naturally acquire through self-supervised learning. This has been investigated especially through probing tasks to analyse the linguistic information that is learned during pre-training of such large models (Liu et al., 2019a; Clark et al., 2019; Kovaleva et al., 2019; Pimentel et al., 2020; Rogers et al., 2020). One type of linguistic information that has been affluently analysed is syntactic information, and most of the recent probing studies have been based on this question: "Can transformer-based large language models learn syntactic structures during pre-training?". Recent studies address this question and propose unsupervised models that use syntactic knowledge obtained from PLMs for NLP tasks such as constituency (Kim et al., 2020a,b; Zeng and Xiong, 2022) and dependency parsing (de Lhoneux et al., 2022).

There are two aims in this study: 1. We aim to analyse the linguistic information that is learned by PLMs in different syntactic levels (dependency, constituency and semantic parsing) which deviates from the previous work, and provide a comparison with different languages. 2. We aim to demonstrate whether it is possible to use the linguistic information learned from PLMs in a fully unsupervised model for dependency, constituency and semantic parsing.

Existing approaches that use pre-trained language models are evaluated mainly on constituency parsing (Kim et al., 2020a,b) and dependency parsing (Hewitt and Manning, 2019; Clark et al., 2019). However, there is not any study that evaluates various parsing levels including semantic parsing using the same parsing model and compares the parsing results to understand the behaviour of PLMs for different levels of parsing from syntax to semantics.

In this paper, we evaluate a fully unsupervised model for three parsing tasks. We adopt the chart-based zero-shot parsing model (Kim et al., 2020b) that is based on the syntactic distance concept (Shen et al., 2017, 2018). To our knowledge, this will be the first study that combines syntactic distance with PLMs to apply to semantic parsing with zero-shot learning. In this study, we particularly use UCCA graph-based semantic representation for semantic parsing, which has been tackled as a constituency parsing problem in previous studies (Jiang et al., 2019; Bölücü and Can, 2021). In addition to the well-studied languages such as English, German, and French, we also evaluate the models for Turkish with a comparably smaller dataset. We obtain the best results with multilingual PLMs. The results show that the zero-shot parsing model performs better with shorter sentences. It also shows that PLMs performs the best with middle layers and the ones closer to the final layers interestingly for all of the three parsing tasks, which are in line with the previous studies.

2 Related Work

A recent research direction has been towards analysing PLMs without fine-tuning on a particular down-stream task to explore the type of information that is learned during pre-training, which is called *probing*. For that purpose, PLMs have been investigated in various tasks such as language modeling (Shen et al., 2017), dependency parsing (Hewitt and Manning, 2019; Clark et al., 2019), constituency parsing (Shen et al., 2017; Peters et al., 2018; Li et al., 2020; Kim et al., 2020b), discourse parsing (Wu et al., 2020), commonsense reasoning (Tikhonov and Ryabinin, 2021), and grammar induction (Shen et al., 2018; Kim et al., 2020a). Some of the studies have also questioned if the syntax is encoded in PLMs (Shi et al., 2016; Blevins et al., 2018; Jawahar et al., 2019) and some of them analysed how large language models encode other types of linguistic information such as coference, entity information, parsing, and semantic roles, and NER (Tenney et al., 2019; Liu et al., 2019a).

In line with our work, Shen et al. (2017) use syntactic distance for character-level and word-level language modeling, and unsupervised constituency parsing. Li et al. (2020) use PLMs for unsupervised constituency parsing focusing on attention heads by ranking and ensembling them. Kim et al. (2020b) propose a model with chart-based decoder for the same problem, which also solves the greedy search problem of Shen et al. (2018). All of these studies are based on the idea that the syntactic structure of sentences are naturally learned along with language modeling. Some of those works (Kim et al., 2020a; Wu et al., 2020) also combine syntactic distance with PLMs to induce syntactic structure in an unsupervised setting. A recent work by Shen et al. (2021a) also introduces joint learning of constituency parsing with dependency parsing in an unsupervised framework.

Our work is similar to these and we also follow the chart-based zero-shot parsing introduced by Kim et al. (2020b). However, this is the first time that several parsing tasks are tackled using the same unsupervised model and this is the first time UCCA-based semantic parsing is performed in an unsupervised setting.

3 Chart-based Zero-shot Parsing

We utilise the syntactic distance concept (Shen et al., 2017, 2018) which was particularly explored for constituency parsing (Kim et al., 2020a,b; Li et al., 2020) by directly using the PLMs without fine-tuning. Here, we adopt chart-based zero-shot parsing based on syntactic distance for three different parsing problems that are semantic, constituency and dependency parsing to explore usability of PLMs in zero-shot setting.

The method calculates scores for spans where an input sentence $s = \{w_1, \cdots, w_n\}$ is made up of a set of labeled spans as follows:

$$T = \{(i_t, j_t, l_t) : t = 1, \cdots, |T|\}$$

where i_t and j_t refer to the beginning and ending positions of the t^{th} span respectively with the label set $l_t \in L$. A score $s(t)$ is assigned to each tree, which is decomposed as follows:

$$s(t) = \sum_{(i,j)\in t} s_{span}(i,j) \qquad (1)$$

Here, $s_{span}(i,j)$ denotes per-span scores that are calculated recursively by splitting spans into smaller spans as defined below:

$$\begin{aligned} s_{split}(i,k,j) &= s_{span}(i,k) + s_{span}(k+1,j) \\ s_{span}(i,j) &= s_{comp}(i,j) + \\ &\quad min_{i\le k<j} s_{split}(i,k,j) \end{aligned}$$

where $s_{comp}(\cdot,\cdot,\cdot)$ measures the validity of the compositionality of the $span(i,j)$ itself, while $s_{split}(i,k,j)$ indicates how plausible it is to split span (i,j) at position k. To calculate $s_{comp}(\cdot,\cdot,\cdot)$, Kim et al. (2020b) introduced two alternative labeled functions. The first one is the characteristic score function $s_c(\cdot,\cdot)$, and the second is the pair score function $s_p(\cdot,\cdot)$. Pair score function computes the average pairwise distance in a given span:

$$s_p(i,j) = \frac{1}{\binom{j-i+1}{2}} \sum_{(w_x,w_y)\in pair(i,j)} f(g(w_x), g(w_y))$$

$$s_c(i,j) = \frac{1}{j-i+1} \sum_{i\le x\le j} f(g(w_x), c)$$

$$c = \frac{1}{j-i+1} \sum_{i\le y\le j} g(w_y)$$

where $pair(i,j)$ returns a set of all combinations of bigrams (e.g. w_x, w_y) inside the span (i,j). Function $f(\cdot,\cdot)$ is the distance measure and $g(\cdot)$ is the representation function. Jensen-Shannon (JSD) and Hellinger (HEL) distance functions are used to measure the distance between two spans. $g = \{g^d_{(u,v)} | u = 1, \cdots, l, v = 1, \cdots, a\}$ returns the v^{th} attention head on the u^{th} layer of the pretrained language model.

CYK (Cocke-Younger-Kasami) (Chappelier and Rajman, 1998) is used for decoding to generate the trees. The parser outputs tree $\hat{t}$ that has the lowest score:

$$\hat{t} = \arg\min_{T} s(t) \tag{2}$$

For each distance function with score functions , we obtain the weights of the i^{th} layer and j^{th} attention head of that layer. Then we calculate the span scores using the distance functions. We select the tree with the lowest score for each distance function, which leads to 4 trees in i^{th} layer and j^{th} attention head. Therefore, we finally obtain $4 \times l \times a$ trees, where l is the number of layers and a is the number of attention heads. The final F1 scores are calculated for each tree and the highest F1 score is reported in the results.

4 Three Levels of Parsing with a Single Model

We use the chart-based zero-shot parsing model for three types of parsing ranging in different semantic and syntactic levels with different granularities and structures of a given text:

Dependency Parsing Dependency parsing is concerned with the syntactic relations between words in a sentence. Those syntactic relations are discovered in terms of dependencies of words on each other. In order to apply the zero-shot parsing model for dependency parsing, we compute the scores for each tree and then we apply Eisner (Eisner, 1996) decoding algorithm (rather than CYK) to produce dependency trees using the tree scores.

Constituency Parsing Constituency parsing is concerned with extracting the syntactic structure of a given text through phrasal constituents. Therefore, it is more concerned with the syntactic structure of an entire sentence rather than the relations between words as opposed to dependency parsing. We apply zero-shot parsing without adding any additional step for constituency parsing.

Semantic Parsing Semantic parsing is concerned with extracting the semantic structure of a given text using a formal representation. We particularly use UCCA (Abend and Rappoport, 2013) graph-based semantic representation to extract semantic relations within the text. In order to perform UCCA-based parsing, we first convert UCCA graphs into constituent trees by removing discontinuities and remote edges (Jiang et al., 2019; Bölücü and Can, 2021). Then we perform zero-shot learning to tackle semantic parsing as a constituency parsing problem. After finding the tree with the lowest score, we convert constituency trees back into the UCCA-based graphs, and restore discontinuity units. We disregard the remote edges and implicit edges.

5 Experiments and Results

We conducted experiments to evaluate the unsupervised parser on dependency, constituency, and semantic parsing for English, German, French, and Turkish since UCCA-annotated datasets are only available for these languages.

5.1 Datasets

- **Dependency Parsing:** We used Universal Dependency v2.3 (Schuster et al., 2017) datasets in English, German, French and Turkish.
- **Constituency Parsing:** We used Penn Treebank (PTB) (Marcinkiewicz, 1994) for English, the SPMRL dataset (Seddah et al., 2013) for German and French, and the Turkish Annotated Treebank (Yıldız et al., 2016) for Turkish.
- **Semantic Parsing:** We used UCCA datasets provided by SemEval 2019 (Hershcovich et al., 2019) in English, German, and French, and the Turkish UCCA-annotated dataset released by Bölücü and Can (2022).

Since it is a zero-shot parsing model and does not involve a training stage, we only used the test sets[1] for all languages for the evaluation.

5.2 Experimental Setting

We use both monolingual and multilingual PLMs in the experiments. For English, we use the following monolingual PLMs: BERT (Devlin et al., 2019),

[1]The details of the datasets are given in Table 7 in Appendix A.

GPT-2 (Radford et al., 2019), RoBERTa (Liu et al., 2019b), and XLNet (Yang et al., 2019). We follow previous work (Kim et al., 2020a,b; Li et al., 2020) by using two variants of each PLM, where the X-base variant consists of 12 layers, 12 attention heads and 768 hidden dimensions, while the X-large variant has 24 layers, 16 attention heads and 1024 hidden dimensions. GPT2 model corresponds to X-base while GPT2-medium corresponds to X-large model.

We use `bert-base-german-cased`, `bert-base-french-europeana-cased`, and `bert-base-turkish-cased` for German, French and Turkish monolingual PLMs respectively.

For multilingual experiments, we use multilingual version of the BERT-base model (M-BERT) (Devlin et al., 2019), the XLM-base model (XLM-R[2]) (Conneau and Lample, 2019), which is a multilingual RoBERTa model, and the large version of XLM (XLM-R-large) (Conneau et al., 2020).

5.3 Results

We present the results obtained from each parsing separately below[4].

Dependency Parsing Dependency parsing results for all languages are given in Table 1. The best results are obtained from multilingual PLMs in all languages. Since the other unsupervised dependency parsing models are either finetuned (Ma and Xia, 2014; Shen et al., 2021b) or utilise other external resources such as Google Universal Treebanks (Ma and Xia, 2014) or WSJ (Shen et al., 2021b), we have not made a comparison with other models since the model presented here is fully unsupervised, does not use any annotated data, and does not incorporate any syntactic information during PLM pre-training.

Constituency Parsing For constituency parsing, we either perform top-down or chart-based parsing to generate trees. We further experiment with using different layers in the PLMs. All unlabeled F1 scores for the constituency parsing are given in Table 2 and Table 3. We use abbreviations TD, CP, and CC for Top-Down, Chart-Pair (pair score function $s_p(\cdot,\cdot)$) and Chart-Characteristic (characteristic score function $s_c(\cdot,\cdot)$) respectively. Except English, we obtain the best results with the top-down decoder and with XLM-R for German, French, and Turkish [5].

Semantic Parsing The unlabeled F_1 scores for UCCA-based semantic parsing are given in Table 4 and 5. The best results are obtained from RoBERTa-base amongst the monolingual models and from XLM-R amongst the multilingual models in English. Interestingly, both RoBERTa and XLM-R gives similar results. For German, French, and Turkish, all the best results are obtained from multilingual models. Since this is the very first study that performs UCCA-based semantic parsing in a completely unsupervised framework, there is not any other study that is available to compare with ours. Therefore, we report our results only as the baseline results for the future studies.

Dependency parsing scores are comparably much lower than both constituency and semantic parsing in all languages. Unsupervised dependency parsing has been mostly performed using probabilistic generative models in the literature (Klein and Manning, 2004) and it is comparatively harder than constituency parsing since it requires learning finer relations between words rather than phrases in a sentence. However, interestingly, UCCA-based semantic parsing scores are also promising and as good as constituency parsing performance. It should be noted that UCCA-based semantic parsing has not been tackled with an unsupervised learning model before.

As for the PLM models, the GPT and GPT2-medium perform comparatively poorly on all parsing problems. Unlike other PLMs, the GPT models are auto-regressive language models that do not allow to incorporate the context on both sides of a word, which might be the reason of the poor performance of the GPT models.

5.4 Analysis of the Results

We analyse the attention layers and heads that contribute the most to each parsing task, along with the affect of the sentence length in the experiments.

5.4.1 Attention Layers

We analyse the attention layers to see which layers provide the most information for the parsing tasks.

[2]The details of the training datasets used in the experiments are given in Table 8 in Appendix B.

[3]We used the pre-trained models of BERT defined in Section A for each language.

[4]We give the results of supervised models in Appendix C.

[5]The model is adopted from that of Kim et al. (2020b) and we prefer not to repeat the comparative scores here again.

Model	*Monolingual Models*			
	English	German	French	Turkish
BERT-base-cased[3]	26.48	26.59	24.78	35.56
BERT-large-cased	27.89	-	-	-
XLNet-base-cased	25.66	-	-	-
XLNet-large-cased	27.53	-	-	-
RoBERTa-base	27.68	-	-	-
RoBERTa-large	25.11	-	-	-
GPT2	19.66	-	-	-
GPT2-medium	21.44	-	-	-
PLM	*Multilingual Models*			
M-BERT	30.80	30.69	**34.37**	**41.62**
XLM-R	30.80	**31.84**	34.27	41.25
XLM-R-large	**32.66**	28.58	26.19	39.13

Table 1: UAS scores for dependency parsing.

	English			German		
Model	TD	CP	CC	TD	CP	CC
	Monolingual Models					
BERT-base-cased	34.51	40.24	42.05	26.96	24.82	26.59
BERT-large-cased	38.93	43.68	44.58	-	-	-
XLNet-base-cased	40.12	42.14	43.47	-	-	-
XLNet-large-cased	38.32	42.60	43.73	-	-	-
RoBERTa-base	40.61	45.37	**46.01**	-	-	-
RoBERTa-large	34.30	42.19	43.26	-	-	-
GPT2	34.21	34.01	35.78	-	-	-
GPT2-medium	37.65	38.59	39.81	-	-	-
	Multilingual Models					
M-BERT	40.28	43.44	44.13	30.69	30.59	30.28
XLM-R	41.25	44.25	44.76	**33.13**	32.19	31.84
XLM-R-large	39.13	42.87	44.67	28.18	27.13	28.58

Table 2: Unlabeled F1 scores for constituency parsing in English and German.

	French			Turkish		
Model	TD	CP	CC	TD	CP	CC
	Monolingual Models					
BERT-base-cased	24.78	22.83	23.86	35.36	31.47	33.50
	Multilingual Models					
M-BERT	32.88	30.37	30.45	41.29	40.61	39.93
XLM-R	**34.19**	31.29	30.93	**45.18**	43.49	42.30
XLM-R-large	26.68	25.70	26.46	36.21	36.72	36.72

Table 3: Unlabeled F1 scores for constituency parsing in French and Turkish.

Dependency Parsing UAS scores obtained from multilingual models for each layer are illustrated in Figure 1. The results show that we get the highest UAS scores from the middle or the ones closer to the final layers of PLMs for all languages.

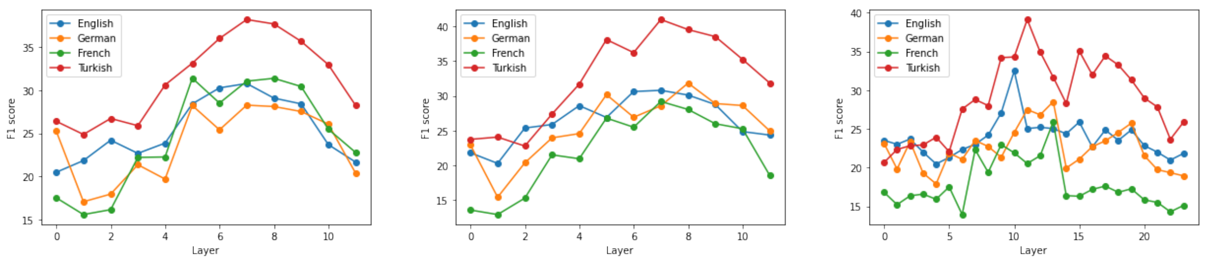
Constituency Parsing F1 scores obtained from multilingual PLMs for all layers are given in Figure 2. Although there are slight differences between languages, the general picture does not differ from the dependency parsing results and again the highest scores are obtained from mostly middle

	English-Wiki			English-20K		
Model	TD	CP	CC	TD	CP	CC
	Monolingual Models					
BERT-base-cased	38.34	42.30	42.60	39.10	42.80	43.93
BERT-large-cased	38.33	42.93	43.52	39.41	43.82	44.75
XLNet-base-cased	37.00	39.62	40.18	37.57	39.56	42.70
XLNet-large-cased	38.41	41.25	42.27	39.98	41.20	41.52
RoBERTa-base	41.82	44.96	**45.21**	32.43	45.62	**46.18**
RoBERTa-large	37.65	41.37	41.62	36.44	41.78	41.92
GPT2	31.97	38.23	38.56	32.41	37.97	38.40
GPT2-medium	34.86	38.49	38.58	32.21	37.68	39.31
	Multilingual Models					
M-BERT	39.62	43.52	44.06	38.11	43.99	45.15
XLM-R	40.98	45.45	**45.89**	42.06	45.51	**46.30**
XLM-R-large	36.40	40.05	40.87	33.69	40.00	41.45

Table 4: Unlabeled F1 scores for semantic parsing in English (English-Wiki, English-20K).

	German-20K			French-20K			Turkish		
Model	TD	CP	CC	TD	CP	CC	TD	CP	CC
	Monolingual Models								
BERT-base-cased[3]	40.30	41.93	42.96	40.32	40.55	42.71	41.49	39.50	42.15
	Multilingual Models								
M-BERT	39.08	**44.17**	44.07	41.01	43.26	46.08	42.15	44.80	44.14
XLM-R	40.90	43.15	42.98	44.13	46.08	**47.38**	46.79	**48.77**	46.79
XLM-R-large	35.59	39.63	42.37	37.56	39.17	38.94	45.46	44.14	46.13

Table 5: Unlabeled F1 scores for semantic parsing in German, French and Turkish.



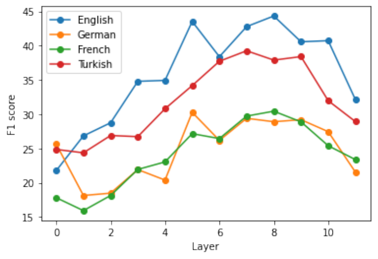
(a) Bert-multilingual (b) XLM-R (c) XLM-R-large

Figure 1: UAS scores of multilingual PLMs for dependency parsing.

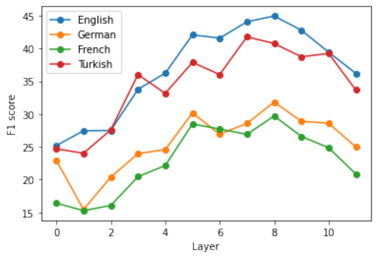
layers.

Semantic Parsing F1 scores obtained from monolingual models for all layers along with different distance functions are given in Figure 3. Only the best scores obtained from the attentions in each layer are illustrated. The graphs show that there is not much difference between the distance functions in terms of their performance in parsing. However, we obtain the highest scores again from the middle or towards the last layers except for GPT-2, which achieves the best in the lower layers.

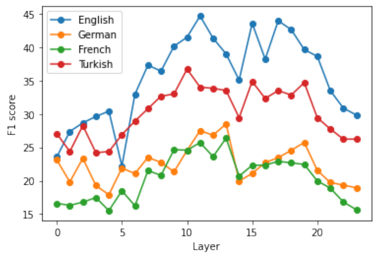
The results obtained from multilingual PLMs for all languages are given in Figure 4. The F1 scores of languages are very low in the first hidden layers except for Turkish. The lower hidden layers might be more informative in short sentences because the Turkish UCCA dataset involves shorter sentences compared to other languages. This might be the reason of such a difference between the languages. The results also support that the final layers bear more syntactic information compared to the lower



(a) Bert-multilingual

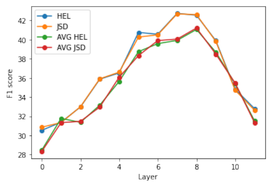


(b) XLM-R

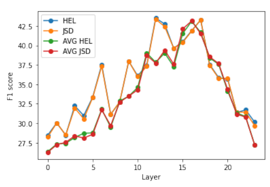


(c) XLM-R-large

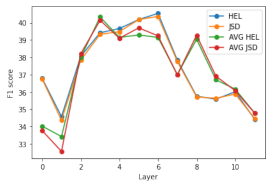
Figure 2: F1 scores of multilingual PLMs for constituency parsing.



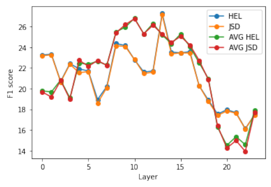
(a) Bert-base



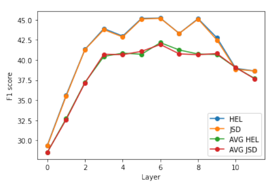
(b) Bert-large



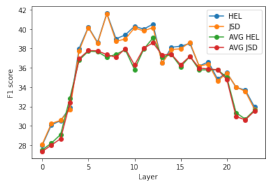
(c) XLNet-base



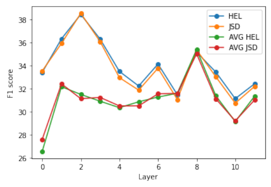
(d) XLNet-large



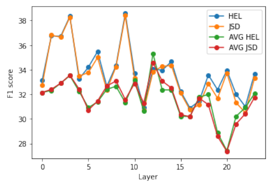
(e) RoBERTa-base



(f) RoBERTa-large

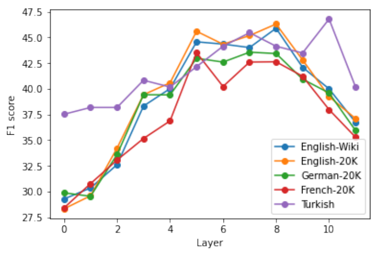


(g) GPT2

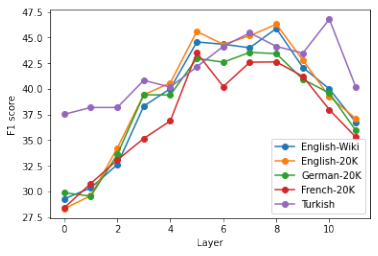


(h) GPT2-medium

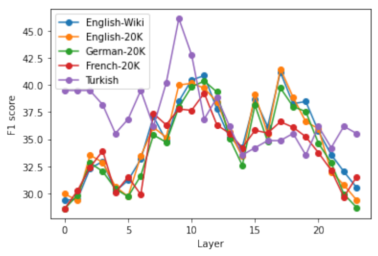
Figure 3: F1 scores from monolingual PLMs using the English Wiki dataset for semantic parsing.



(a) Bert-multilingual

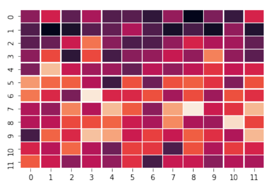


(b) XLM-R

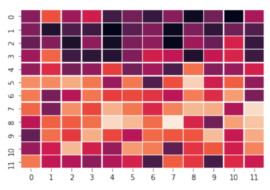


(c) XLM-R-large

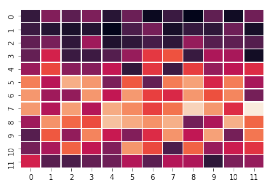
Figure 4: F1 scores from multilingual PLMs using the UCCA datasets.



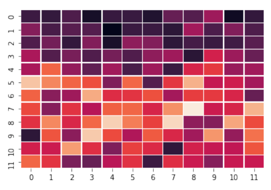
(a) English



(b) German



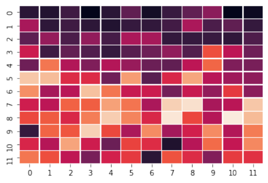
(c) French

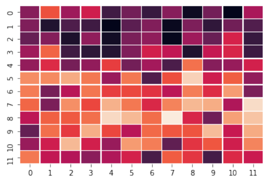


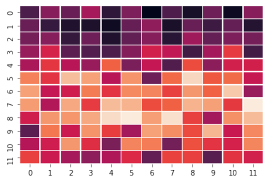
(d) Turkish

Figure 5: Unsupervised dependency parsing performance in all languages according to different attention heads and hidden layers with HEL distance function (Light cells refer to higher UAS scores).

layers, especially in longer sentences, which is consistent with the findings of other studies (Clark et al., 2019; Kim et al., 2020b,a).


(a) English


(b) German


(c) French

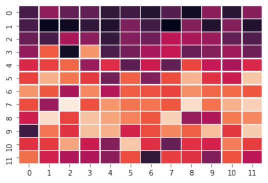
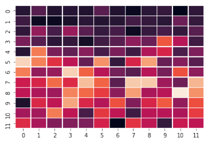
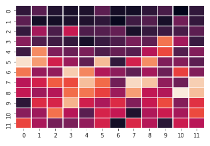
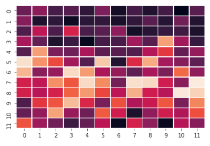
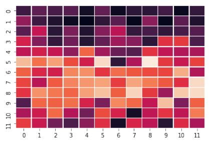

(d) Turkish

Figure 6: Unsupervised constituency parsing performance in all languages for different attention heads and hidden layers with HEL distance function (Light cells correspond to higher F1 scores).


(a) English-Wiki


(b) English-20K


(c) German-20K


(d) French-20K

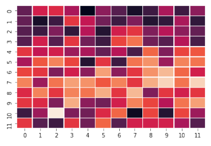

(e) Turkish

Figure 7: Unsupervised UCCA semantic parsing performance in all languages with different attention heads and hidden layers with HEL distance function (Light cells refer to higher F1 scores).

5.4.2 Attention Heads

We also analyse the attention heads in the layers to observe which attention heads contribute the most to each parsing task. F1 scores obtained from the attention heads in different layers are given in Figure 5, Figure 6, and Figure 7 for dependency, constituency, and semantic parsing (with XLM-R) respectively. The graphs support the findings regarding the hidden layers and further show that top heads contain more information in all tasks and languages apart from Turkish constituency and semantic parsing where the lower heads contain more information. This might be again due to the length of the sentences in the Turkish datasets.

5.4.3 Sentence Length

To understand the effect of the sentence length, we extract the average length of the sentences in all datasets. The average sentence length of Turkish datasets for all tasks is less than that of the other languages, whereas the average sentence length of German and French is higher in all parsing datasets.

To investigate the relationship between the sentence length and the accuracy of the parsing, we run the constituency parsing with XLM-R multilingual PLM and top-down parser on 1000 samples with a length less than the average length of the dataset and 1000 samples with a length greater than the average length of the datasets in English, French and German. We only use 50 samples (25 less and 25 are greater than the average length in Turkish since there are only 63 samples in the dataset. Table 6 gives the average length of the sentences in each dataset along with the obtained F1 scores. The results show that the model performs better on shorter sentences. This also confirms that the model can hardly find distant relationships in longer sentences.

6 Conclusion

We analyse the syntactic information learned by transformer-based PLMs for various parsing problems (namely dependency, constituency, and semantic parsing) using a fully unsupervised zero-shot parser. To the best of our knowledge, this is the first study that compares an unsupervised model for three different parsing problems in a fully unsupervised setting and analyses the linguistic information learned from PLMs during pre-training for three different parsing tasks from syntax to semantics. The results show that PLMs provide information from mostly middle and towards the final layers for all parsing tasks, which is also in line with the previous work on constituency and dependency parsing. However, interestingly, the study shows that when it comes to structure learning, syntax and semantics are both encoded in middle and towards the final layers.

References

Omri Abend and Ari Rappoport. 2013. UCCA: A semantics-based grammatical annotation scheme. In *Proceedings of the 10th Interna-*

		samples< average		samples>average	
Language	Av. Len	Av. Len	F1	Av. Len	F1
English	20.46	12.76	43.73	28.96	39.04
German	18.82	11.62	33.71	30.07	30.37
French	29.73	16.82	34.64	46.05	31.85
Turkish	13.95	11.84	48.42	16.48	44.29

Table 6: F1 scores of constituency parsing for samples that are shorter and longer than the average overall sentence length.

tional Conference on Computational Semantics (IWCS 2013)–Long Papers, pages 1–12.

Terra Blevins, Omer Levy, and Luke Zettlemoyer. 2018. Deep RNNs encode soft hierarchical syntax. *CoRR*, abs/1805.04218.

Necva Bölücü and Burcu Can. 2022. Turkish universal conceptual cognitive annotation. In *Proceedings of LREC 2022*. European Language Resources Association.

Necva Bölücü and Burcu Can. 2021. Self-attentive constituency parsing for UCCA-based semantic parsing. *arXiv preprint arXiv: 2110.00621*.

J-C Chappelier and Martin Rajman. 1998. A generalized CYK algorithm for parsing stochastic CFG. In *Proceedings of the 1st Workshop on Tabulation in Parsing and Deduction (TAPD'98)*, CONF, pages 133–137.

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.

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.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. *Advances in neural information processing systems*, 32.

Miryam de Lhoneux, Sheng Zhang, and Anders Søgaard. 2022. Zero-shot dependency parsing with worst-case aware automated curriculum learning. *arXiv preprint arXiv:2203.08555*.

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.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1127–1138.

Daniel Hershcovich, Zohar Aizenbud, Leshem Choshen, Elior Sulem, Ari Rappoport, and Omri Abend. 2019. SemEval-2019 Task 1: Cross-lingual semantic parsing with UCCA. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 1–10.

John Hewitt and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138.

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*.

Wei Jiang, Zhenghua Li, Yu Zhang, and Min Zhang. 2019. HLT@ SUDA at SemEval-2019 Task 1: UCCA graph parsing as constituent tree parsing. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 11–15.

Taeuk Kim, Jihun Choi, Sang-goo Lee, and Daniel Edmiston. 2020a. Are Pre-trained Language Models Aware of Phrases? Simple but Strong Baselines for Grammar Induction. In *International Conference of Learning Representations*.

Taeuk Kim, Bowen Li, and Sang-goo Lee. 2020b. Multilingual chart-based constituency parse extraction from pre-trained language models. *arXiv preprint arXiv:2004.13805*.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 478–485, Barcelona, Spain.

Daniel Kondratyuk. 2019. 75 languages, 1 model: Parsing universal dependencies universally. *CoRR*, abs/1904.02099.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. *arXiv preprint arXiv:1908.08593*.

Bowen Li, Taeuk Kim, Reinald Kim Amplayo, and Frank Keller. 2020. Heads-up! unsupervised constituency parsing via self-attention heads. 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 409–424, Suzhou, China.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. *CoRR*, abs/1903.08855.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Xuezhe Ma and Fei Xia. 2014. Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1337–1348.

Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Using Large Corpora*, page 273.

Matthew Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Brussels, Belgium. Association for Computational Linguistics.

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. 2020. Information-theoretic probing for linguistic structure. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4622, Online.

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.

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. *arXiv preprint arXiv:1910.01108*.

Sebastian Schuster, Matthew Lamm, and Christopher D. Manning. 2017. Gapping constructions in universal dependencies v2. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 123–132, Gothenburg, Sweden.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, et al. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182. Association for Computational Linguistics.

Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron Courville. 2017. Neural language modeling by jointly learning syntax and lexicon. *arXiv preprint arXiv:1711.02013*.

Yikang Shen, Zhouhan Lin, Athul Paul Jacob, Alessandro Sordoni, Aaron Courville, and Yoshua Bengio. 2018. Straight to the tree: Constituency parsing with neural syntactic distance. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1180, Melbourne, Australia. Association for Computational Linguistics.

Yikang Shen, Yi Tay, Che Zheng, Dara Bahri, Don Metzler, and Aaron Courville. 2021a. Structformer: Joint unsupervised induction of dependency and constituency structure from masked language modeling. In *ACL 2021*.

Yikang Shen, Yi Tay, Che Zheng, Dara Bahri, Donald Metzler, and Aaron Courville. 2021b. StructFormer: Joint unsupervised induction of dependency and constituency structure from masked language modeling. 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 7196–7209, Online.

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.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. *CoRR*, abs/1905.05950.

Alexey Tikhonov and Max Ryabinin. 2021. It's all in the heads: Using attention heads as a baseline for cross-lingual transfer in commonsense reasoning. *arXiv preprint arXiv:2106.12066*.

Zhiyong Wu, Yun Chen, Ben Kao, and Qun Liu. 2020. Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4166–4176, Online.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Olcay Taner Yıldız, Ercan Solak, Şemsinur Çandır, Razieh Ehsani, and Onur Görgün. 2016. Constructing a Turkish constituency parse treebank. In *Information Sciences and Systems 2015*, pages 339–347. Springer.

Zhiyuan Zeng and Deyi Xiong. 2022. Unsupervised and few-shot parsing from pretrained language models. *Artificial Intelligence*, page 103665.

A Details of the test sets

Here specify the size of the test sets used in all parsing tasks.

	English	German	French	Turkish
DP	2077	1000	416	979
CP	2416	5000	2541	63
SP	Wiki: 515 20-K: 492	652	239	50

Table 7: Size of the test sets used in the experiments (DP: Dependency parsing, CP: Constituency parsing, and SP: Semantic parsing)

B Details of the training sets for XLM-R

Here we present the size of the monolingual datasets used for training the XLM-R.

Language	Tokens (M)	Size (GiB)
English	55608	300.8
German	10297	66.6
French	9780	56.8
Turkish	2736	20.9

Table 8: Size of each monolingual dataset used for training the XLM-R.

C Supervised model results for three levels of parsing

Here we give the results obtained from supervised models for dependency and semantic parsing problems with the best results of the unsupervised model in the paper[6].

Model	English	German	French	Turkish
Our Model	32.66	31.84	34.37	41.62
UDPipe ♣	89.63	85.53	90.65	74.19
UDify ♣	90.96	87.81	93.60	74.56

Table 9: Comparative UAF scores of our unsupervised model with supervised models for dependency parsing (♣: Kondratyuk (2019))

[6]We couldn't give the constituency parsing results since the studies on constituency parsing present only labeled scores.

[7]We used the zero-shot experimental results in the paper of (Bölücü and Can, 2022) for Turkish dataset.

Model	English-Wiki	English-20K	German-20K	French-20K	Turkish
Our Model	45.89	46.30	44.17	47.38	48.77
Tupa ♣	85.00	82.20	90.30	74.00	-
HLT@SUDA ♡	87.20	85.20	92.80	86.00	-
Self-Attentive ♠	89.60	87.69	94.10	86.00	76.80[8]

Table 10: Comparative unlabeled F-1 scores of our unsupervised model with supervised models for semantic parsing (♣: Hershcovich et al. (2017), ♡: Jiang et al. (2019), ♠: Bölücü and Can (2021))

Knowledge Enhanced Deep Learning Model for Radiology Text Generation

Kaveri Kale[1], **Pushpak Bhattacharyya**[1],
Aditya Shetty[2], **Milind Gune**[3],
Kush Shrivastava[4], **Rustom Lawyer**[4] and **Spriha Biswas**[4]

[1]Department of Computer Science and Engineering, IIT Bombay, India
[2]Consultant Radiologist, Breach Candy Hospital, India
[3]Consultant Radiologist, Thane, India
[4]Augnito India Private Limited

{kaverikale,pb}@cse.iitb.ac.in,
adityashetty01@gmail.com, dgune@rediffmail.com, {kush.shrivastava, rustom, spriha}@augnito.ai

Abstract

Manual radiology report generation is a time-consuming task. First, radiologists prepare brief notes while carefully examining the imaging report. Then, radiologists or their secretaries create a full-text report that describes the findings by referring to the notes. Automatic radiology report generation is the primary objective of this research. The central part of automatic radiology report generation is generating the finding section (main body of the report) from the radiologists' notes. In this research, we suggest a knowledge graph (KG) enhanced radiology text generator that can provide additional domain-specific information. Our approach uses a KG-BART model to generate a description of clinical findings (referred to as **pathological description**) from radiologists' brief notes. We have constructed a parallel dataset of radiologists' notes and corresponding pathological descriptions to train the KG-BART model. Our findings demonstrate that, compared to the BART-large and T5-large models, the BLEU-2 score of the pathological descriptions generated by our approach is raised by 4% and 9%, and the ROUGE-L score by 2% and 2%, respectively. Our analysis shows that the KG-BART model for radiology text generation outperforms the T5-large model. Furthermore, we apply our proposed radiology text generator for whole radiology report generation.

1 Introduction

Due to the meager ratio of radiologists to patients, radiologists are in high demand. The ratios in the US, China and India are 1:10,000, 1:14,772, and 1:100,000 respectively (Arora, 2014). It leads to a large influx of patients, which keeps radiologists extremely busy and under stress. To boost the effectiveness and productivity of radiologists, several hospitals and diagnostic facilities have established radiology information systems (RIS) and picture archiving and communications systems (PACS) (Honeyman, 1999). Despite this, the current workflow causes a delay in the turnaround time for reports, report inaccuracies, and burnout. Our conversations with radiologists have revealed that many radiologists wish to eliminate the tiresome report-generating process and concentrate on the diagnosis.

The main task in automatic radiology report generation is generating pathological descriptions from radiologists' notes. In Natural Language Processing (NLP), we can look at this as a text generation task. Various neural encoder-decoder models have been proposed to accomplish the text generation goal by learning to map input text to output text. However, the input text alone often provides limited knowledge to generate the desired output. Hence, researchers have considered incorporating external knowledge from domain-specific KGs along with internal knowledge embedded in the input text (Yu et al., 2022). KG-BART (Liu et al., 2021) model incorporates the domain-specific KG in the deep learning model. In our work, we incorporate ultrasound radiology KGs (Kale et al., 2022) in the KG-BART model to generate radiology text from radiologists' input notes. We construct a radiology domain dataset to train the KG-BART model. We obtain grounded KG for input sentences. KG grounding finds the most relevant entities and relations from radiology KG to guide the KG-BART model to better understand the relationships among concepts. It considers the inter-concept relation and significant neighbor entities to generate a more natural and plausible output. Two high-profile radiologists are associated with this research who help us to get domain insights and to create a dataset.

Our contributions in this paper are as follows:

- Parallel dataset of radiologists' notes and corresponding pathological descriptions.

- Our work shows that the KG-BART is strong choice for radiology text generation than other state-of-the-art models like T5-base/large, BART-base/large.
- Our KG grounding approach reduces noise (irrelevant neighbor entities) and obtains the most relevant neighbor entities.

2 Background Concept and Terminology

Traditionally, radiologists either dictate on a voice recorder or write the diagnosis notes (referred to as **radiologist's notes**) on paper. Their secretaries are then given access to the notes. Next, the secretary access a normal report template, which is a scan-specific normal template (referred to as **normal report template**) that corresponds to all normal findings, and creates a preliminary report by altering it in accordance with the measurements and findings that the radiologist reported in a more detailed form (**pathological description**). The radiologist receives the preliminary report once again. The radiologist then reads the report, makes any necessary corrections, and then signs off. Finally, the report is provided to a doctor or patient for potential follow-up care. Table 1 shows the examples of radiologists' notes and corresponding pathological descriptions. The average number of words in radiologists' notes is 15, and the average number of words in pathological descriptions is 26.

Radiologists' Notes	**Pathological Description**
Normal uterus 1 x 5 x 3.4 mm with hypoechoic fibroid 2.3 x 5.6 mm in fundic body region.	Uterus is anteverted and normal in size 1 x 5 x 3.4 mm. Myometrial reflectivity is inhomogeneous and shows a hyperechoic fibroid in fundic body region measuring 2.3 x 5.6 mm.
Liver shows generalized fatty infiltration severe hepatomegaly noted.	Liver severely enlarged and it reveals generalized fatty infiltration.
Cirrhosis with portal hypertension 6 cm.	Liver is small and shrunken and coarse echotexture margin are nodular. portal vein is mildly dilated, it measures 6 cm.

Table 1: Examples of radiologists' notes and corresponding pathological descriptions.

In order to create an accurate diagnostic description from brief notes, domain-specific knowledge will be helpful, given that the knowledge graph can supply relationship information to strengthen the capacity for reasoning and provide adjunct entities to the concept. In our work we use radiology (ultrasound) KGs constructed by Kale et al. (2022). These KGs are constructed for each organ separately. Since all KGs are hierarchical, and the root of each KG is an organ name (e.g., *liver, gallbladder, pancreas*, *etc.*), we have integrated all these KGs into a single KG (reffered as **ultrasound KG**) by adding *upper abdomen* as root entity. First we extract the grounded KGs for each input concept set from ultrasound KG and then we incorporate grounded KGs in KG-BART model to generate pathological descriptions. The **KG grounding** is the process of extracting the subgraphs (referred to as grounded KGs) from domain-specific KG (in our case ultrasound KG). A grounded KG is a subgraph from the KG whose nodes are concepts in the input plus additional nodes. While doing KG grounding we construct two graphs, i) Input-concept graph and ii) Concept-expansion graph. The expansion is due to the KG, supplying related concepts closely related to those in the input. **Input-concept Graph:** It consists of (a) nodes in the KG matching with input concepts and (b) nodes that are along the paths to the root node of the KG, containing these nodes. **Concept-expansion Graph:** It is the input-concept graph plus the relevant children of the nodes in the input-concept graph.

For input-concept graph and concept-expansion graph, we encode the entity representations and their dependency relations using Knowledge Graph Embeddings (KGE) (Choudhary et al., 2021). KGE represents the entities and relations in lower-dimensional vectors that can be efficient for computations.

3 Related Work

With prior knowledge of chest findings, Zhang et al. (2020) created a graph model that could be used in deep learning models. Disease findings are represented in this network as nodes, and related findings are connected closely between them so that they might impact one another during the propagation and aggregation of the graph. To learn specific attributes for each node in the graph, they incorporate this graph into the deep neural network. Features extracted from KG are used for multilabel classification. To improve pre-training language understanding, Zhang et al. (2019) integrates KG instructive items that are contextually aligned. KEPLER uses a pre-trained language understanding model to encode textual descriptions of entities before integrating the goals of knowledge embedding and language modeling (Wang et al., 2021). By including triples from the KG as supplemental words, K-BERT infuses domain information into the models (Liu et al., 2020). In light of these studies, we contend that additional knowledge data can significantly improve the performance of pre-training models used for text generation tasks.

4 Dataset Construction

We fetch impressions and corresponding pathological descriptions from the radiology text report corpus to construct a parallel dataset. The radiology text report corpus contains anonymized radiology ultrasound reports that are provided by a company collaborating with us, with due consent of the physicians. We have approximately 10 lac radiology reports, out of which around 1 lac reports are of ultrasound. The radiology report contains the title, history, findings, and impression sections. Each section's content is well-structured despite not being uniform. As a result, we use heuristics like regular expressions and word overlap to identify different sections. Each section's content is tokenized and lower-cased. Impressions are very close to radiologists' notes, but it does not contain patient-specific information like measurement of findings, anatomical location, *etc.* To convert impressions into radiologists' notes, we manually edit impressions by adding patient-specific information like measurements and anatomical locations by referring to pathological descriptions. Examples of impressions, their corresponding pathological descriptions and radiologists' notes prepared using impressions are given in table 2.

4.1 Data Preprocessing

Reports contain free-text clinical narratives. Therefore it has many spelling mistakes and writing mistakes as well. We perform the following preprocessing tasks on a parallel dataset:

- In the corpus, there are a lot of extra spaces and unwanted punctuation marks found. We remove these unwanted characters from the corpus using regular expressions.
- We apply sentence segmentation to identify sentence boundaries between different sentences.
- We use SymSpell[1] library to correct the spellings by applying the unigram and bigram dictionaries. We create dictionaries from the corpus and correct them manually.

Once the data cleaning process done. Our domain experts verify the dataset manually and correct it if necessary. We try to create a dataset because the radiology dataset for ultrasound is not publicly available.

4.2 Concept Extraction

KG-BART model needs input and target dataset for training and validation. We give text input to the KG-BART model in the form of a concept set. **Concept set** is the set of radiological entities extracted from radiologist's notes. For example radiological concepts present in note, *Lesion found in right lobe of liver.* are *lesion, right lobe,* and *liver*. To extract the concepts from radiologists' notes, we use an entity extractor based on the method explained in the paper (Kale et al., 2022). Table 2 shows the examples of radiologists' notes and concept sets extracted from notes along with their corresponding pathological descriptions.

5 Method

Figure 1 shows the architecture of the text-generation model with input/output flow. The main components of our model are KG grounding, KG embeddings, text embedding, encoder, and decoder. This section explains all these components in detail.

5.1 Knowledge Graph Grounding

We extract the small subgraphs, input-concept, and concept-expansion graphs for each input sample from our ultrasound KG.

Algorithm to construct input-concept graph and concept-expansion graph is given below: i) For each input concept set in our dataset, we link input concepts with ultrasound KG entities using entity matcher. Entity matcher implemented using TheFuzz[2] library. ii) To find the appropriate path in the ultrasound KG. First, we find all possible candidate paths from matched entity to the root node. We find the most appropriate top five paths by using ranking based on precision and recall of entities in concept set and entities in all possible candidate paths. We consider all paths which include matched entity which is absent in the already selected top-ranking path. iii) Our algorithm constructs an input-concept

[1]https://symspellpy.readthedocs.io

[2]https://github.com/seatgeek/thefuzz

Impression	Radiologists' Notes	Concept Set	Pathological Description
Bulky retroverted uterus with fundal fibroid.	Bulky retroverted uterus with fundal fibroid 2.3 x 5.6 mm.	uterus, fibroid, bulky, fundal, retroverted, 2.3 x 5.6 mm	Uterus is retroverted and bulky in size. Myometrial reflectivity is inhomogeneous with an illdefined fundal fibroid measuring 2.3 x 5.6 mm noted.
Calculus cholecystitis with multiple large calculi.	Calculus cholecystitis with multiple large calculi within lumen of gallbladder, largest measuring 2.4 mm.	multiple, calculi, calculus, lumen, cholecystitis, enlarged, measuring, 2.4 mm	Gallbladder is distended reveals thick wall. Feature of note is presence of multiple large calculi seen within lumen of gallbladder; largest calculus measures 2.4 mm.
Acute pancreatitis.	Acute pancreatitis.	acute pancreatitis	Pancreas is bulky, reveals reduced reflectivity with increased reflectivity of peripancreatic fat.

Table 2: Samples from dataset constructed using radiology report corpus. The first column shows the impressions extracted from the radiology report, and the last column shows the pathological description corresponding to the impression fetched from the radiology report. The second column shows the radiologists' notes prepared by adding patient-specific information to the impression. The third column shows the concepts extracted from radiologists' notes. The final training dataset contains only concept set (as input) and pathological description (as target) columns.

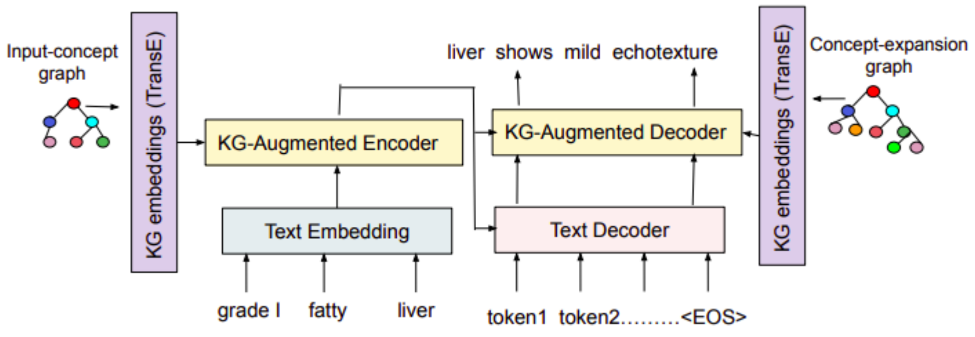


Figure 1: Our model architecture with input as radiologist's dictation and output as pathological description.

Total Samples	Train Samples	Test Samples	Validation Samples
6860	6000	430	430

Table 3: Statistics of the parallel dataset. Training dataset contains concept sets and corresponding pathological descriptions.

and concept-expansion graphs containing all paths that we have selected using a ranking algorithm and neighbor nodes which are the default properties of node present in path. Since ultrasound KG is hierarchical KG where if the node is **finding**, then its parent is the **anatomical location**, and its children are **properties** of findings. Hence, even if some information is missing in the input, we can get it from the input-concept or concept-expansion graphs.

Algorithm 1 gives the pseudocode to construct input-concept and concept-expansion graphs. Adding one-hop, two-hop, or n-hop neighbors of concept nodes adds irrelevant nodes in the expanded graph, which leads to noise. Our approach reduces the noise and obtains the most relevant neighbor nodes. Instead of passing these graphs (as it is for training), the model represents it in vector form. KG embedding module produces the vector representation for input-concept and concept-expansion graphs. Example of the input-concept and concept-expansion graphs is shown in figure 2.

5.2 KG Embeddings

The ultrasound KG is represented in low dimensional vector space using KGE. For simplicity and concreteness, in this work, we primarily consider TransE (Bordes et al., 2013) model due to their state-of-the-art performance. To implement the TransE model for KG embeddings, we use the open-source OpenKE[3] tool. Ultrasound KG contains 860 nodes and 1016 triples. Triples are divided into train and validation sets. The training triple set contains 900 triples, and the validation triple set contains 116 triples.

[3]https://github.com/thunlp/OpenKE

Algorithm 1: Construct input-concept and concept-expansion graphs.

```
Input  : CS: Concept Set
         G(V, E) : Knowledge Graph
Output : Input-concept and concept-expansion graph
1  Find all candidate paths in G(V, E) that includes the node with input concept
2  path-dict -> initialize
3  for each path in possible candidate-paths do
4      Precision = $\frac{CS \cap \text{All entities in path}}{\text{No. of concepts in CS}}$
5      Recall = $\frac{CS \cap \text{All entities in path}}{\text{No. of nodes in path}}$
6      F-score = $\frac{2*\text{Precision}*\text{Recall}}{\text{Precision}+\text{Recall}}$
7      add path-dict -> (path:F-score)
8  end
9  Sort path-dict in descending order of F-score
10 Get top 5 paths
11 for each path in top-5-paths do
12     if len(set($CS$) - set(path)) > 0 then
13         Add all triplets from that path in input-concept graph triplet set
14         Add all triplets from that path in concept-expansion graph triplet set
15         for each node in path do
16             Find all neighbors of node with default-property relation
17             Add all triples of form (neighbor, DefaultPropertyOf, node) in concept-expansion graph triplet set
18         end
19     end
20 end
21 Save input-concept graph triples set (input-concept graph) in csv file.
22 Save concept-expansion graph triples set (concept-expansion graph) in csv file.
```

5.3 Text Embedding

The input embeddings are made of two separate embeddings, 1) Token embeddings and ii) Position embeddings. To get the final text embedding we add the vectors of token embeddings and position embeddings.

5.3.1 Token Embeddings

Tokens are nothing but a word or a part of a word. The textual encoder uses the vocabulary offered by large-cnn BART, and the token embedding is consistent with BART. Using a trainable lookup table, we transform each token in the input concept-set into an embedding vector.

In order to create these token embeddings, a method called BART tokenizer is used to tokenize the text. The encoder, decoder, and language modeling head (Press and Wolf, 2016) all share the embedding parameters. Due to the permutation-invariance of the attention layers, BART learns positional embeddings for absolute token positions and adds them to the token embeddings (Vaswani et al., 2017; Devlin et al., 2018).

5.3.2 Positional Embeddings

Position embeddings represents the position of the word within that sentence that is encoded into a vector. We must introduce some information about the relative or absolute location of the tokens in the sequence because our model lacks recurrence and convolution and hence cannot use the sequence's order. To do this, we augment the token embeddings at the base of the encoder and decoder stacks with positional embeddings. The text embeddings are the sum of the token embeddings and the positional embeddings.

5.4 Encoder

The encoder uses two modalities- text, and KG to condition the generation. According to Figure 1, the KG enhanced encoder layer sits above the text embedding layer and is intended to enhance the text representation by taking the KG structure into account. We use a graph attention layer to incorporate graph representations into the input encoding process. It uses explicit relations to help the model learn intra-concept relations more effectively. Formally, the grounded KG embedding, as well as the text embeddings, are combined by the KG-augmented encoder to update the text token representation. Our self-attention layer and fully-connected layer with residuals make up the stack of m transformer blocks that make up our bidirectional KG-augmented encoder.

5.5 Decoder

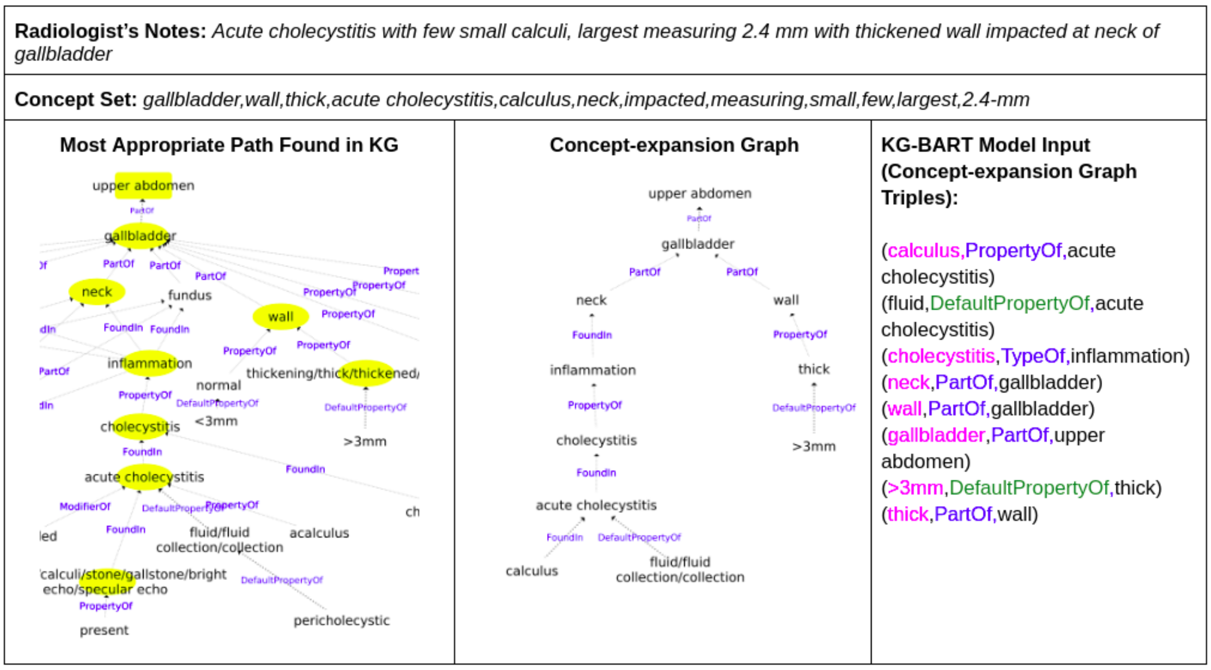
The decoder uses the text embedding module at the bottom layer to encode the text. Similar to encoder, decoder contains KG-augmented decoder layer. It incorporates a concept-expansion graph to get input concepts' missing information and context. The decoder of our model is also a multi-layer transformer. Our decoder is auto-regressive and unidirectional. We skip over a detailed explanation of these modules because our textual transformers are the same as those used in BART (Lewis et al., 2019) and (Vaswani et al., 2017).

6 Experimental Setup

The model input consists of the concept set and KG encoding for the input-concept and concept-expansion graphs. The output is the TARGET statement, i.e., pathological description. We use above

	BLEU Score				ROUGE Score								
					1-gram			3-gram			L-gram		
	1-gram	**2-gram**	**3-gram**	**4-gram**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
T5-base	0.81	0.74	0.68	0.63	0.87	0.88	0.87	0.76	0.77	0.76	0.84	0.85	0.84
T5-large	0.85	0.80	0.75	0.72	0.93	0.88	0.90	0.84	0.8	0.81	0.92	0.87	0.89
BART-base	0.86	0.82	0.78	0.75	0.93	0.90	0.91	0.84	0.82	0.83	0.91	0.89	0.90
BART-large	<u>0.89</u>	<u>0.85</u>	<u>0.84</u>	<u>0.81</u>	<u>0.93</u>	<u>0.92</u>	<u>0.92</u>	<u>0.86</u>	<u>0.86</u>	<u>0.86</u>	<u>0.93</u>	<u>0.92</u>	<u>0.92</u>
KG-BART	**0.93**	**0.89**	**0.86**	**0.83**	**0.96**	**0.96**	**0.95**	**0.89**	**0.89**	**0.88**	**0.94**	**0.94**	**0.93**

Table 4: BLEU and ROUGE score of generated pathological descriptions by T5-base/large, BART-base/large, and KG-BART models vs. gold standard pathological descriptions. The best results are in bold font, and the second best is underlined.


Radiologist's Notes: *Acute cholecystitis with few small calculi, largest measuring 2.4 mm with thickened wall impacted at neck of gallbladder*

Concept Set: *gallbladder,wall,thick,acute cholecystitis,calculus,neck,impacted,measuring,small,few,largest,2.4-mm*

KG-BART Model Input (Concept-expansion Graph Triples):

(calculus,PropertyOf,acute cholecystitis)
(fluid,DefaultPropertyOf,acute cholecystitis)
(cholecystitis,TypeOf,inflammation)
(neck,PartOf,gallbladder)
(wall,PartOf,gallbladder)
(gallbladder,PartOf,upper abdomen)
(>3mm,DefaultPropertyOf,thick)
(thick,PartOf,wall)

Figure 2: Left hand side graph is the snapshot of ultrasound KG that we are using for training. Nodes highlighted in yellow shows the concepts from the concept set that matches with the KG entities. Right hand side graph is the concept-expansion graph constructed for input concept set.

mentioned constructed dataset to train our model. Table 3 shows the statistics of constructed dataset.

6.1 Retraining Setup

We have implemented our own algorithm for KG-grounding task. We use pre-trained KG-BART[4] model which was trained for commonsense reasoning on ConceptNet KG and commonsense dataset. We fine tune this model on radiology text dataset that we have constructed. We use byte-pair encoding for tokenization with a maximum length of 32 for the encoder and 64 for the decoder. We set learning rate to 0.00001 and used AdamW with 1 = 0.9, 2 = 0.98 for optimization. We set the batch size to 32. We trained the KG-BART for 15 epochs, and the gradients are accumulated every 6 steps. We apply dropout with a probability 0.1 to avoid over-fitting. We use beam search with beam size 5 and length penalty with factor 0.6 while inferencing. The training time took 7 hrs on a single NVIDIA GeForce GTX 1080 Ti GPU with 11 GB GDDR5X memory.

7 Baseline and Evaluation

We compare the performance of KG-BART model with T5-base/large (Raffel et al., 2020) and BART-base/large (Lewis et al., 2019) state-of-the-art pre-trained conditional text generation models. Following other conventional generation tasks, we use several widely-used automatic metrics to automatically assess the performance, such as BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004), which mainly focus on measuring n-gram similarities. Ta-

[4]https://github.com/yeliu918/KG-BART

ble 4 shows the BLEU score and ROUGE score of generated pathological descriptions by KG-BART and T5-base/large and BART-base/large models vs. gold standard pathological descriptions.

8 Qualitative Analysis

KG-BART model performs better on unseen data. Sentence formation of the KG-BART model is better than T5 and BART models when input is in abstract form and random in sequence. As shown in table 5, the output generated for input one by the BART-large model does not consider the extra part of notes which does not occur in the training set. Also, in most cases like example two, the KG-BART model correctly identifies the finding location since KG-BART gets the hierarchical anatomical location information from the KG.

9 Radiology Report Generation Using KG-BART Radiology Text Generator

Radiology report generation includes following main tasks:

- Generate pathological description from radiologists' notes.
- Replace appropriate normal sentences (referred as **normal description**) in normal report template with generated pathological descriptions.

For the first task we use our proposed radiology text generator to generate pathological description from radiologists' notes. This section gives the details of second task; how to replace generated pathological description in normal report template to generate whole report. Our domain experts provide us normal report templates. For example, Male Abdomen Pelvis Ultrasound Normal Report, Female Abdomen Ultrasound Normal Report *etc.* According to patient's gender and scan procedure we provide an appropriate normal report template to the system. System finds the appropriate normal sentences to replace with generated pathological description and replace it. As we discussed with hospitals, radiologists, physicians, *etc.*, they are happy to provide impression by themselves to generate whole report. We add impressions provided by radiologists in impression section and generates the whole report.

9.1 Replace Appropriate Normal Sentences with Generated Pathological Descriptions

We create a parallel corpus for the radiologists' notes and the corresponding normal descriptions. Table 6 shows the samples from the parallel corpus of radiologists' notes and normal descriptions.

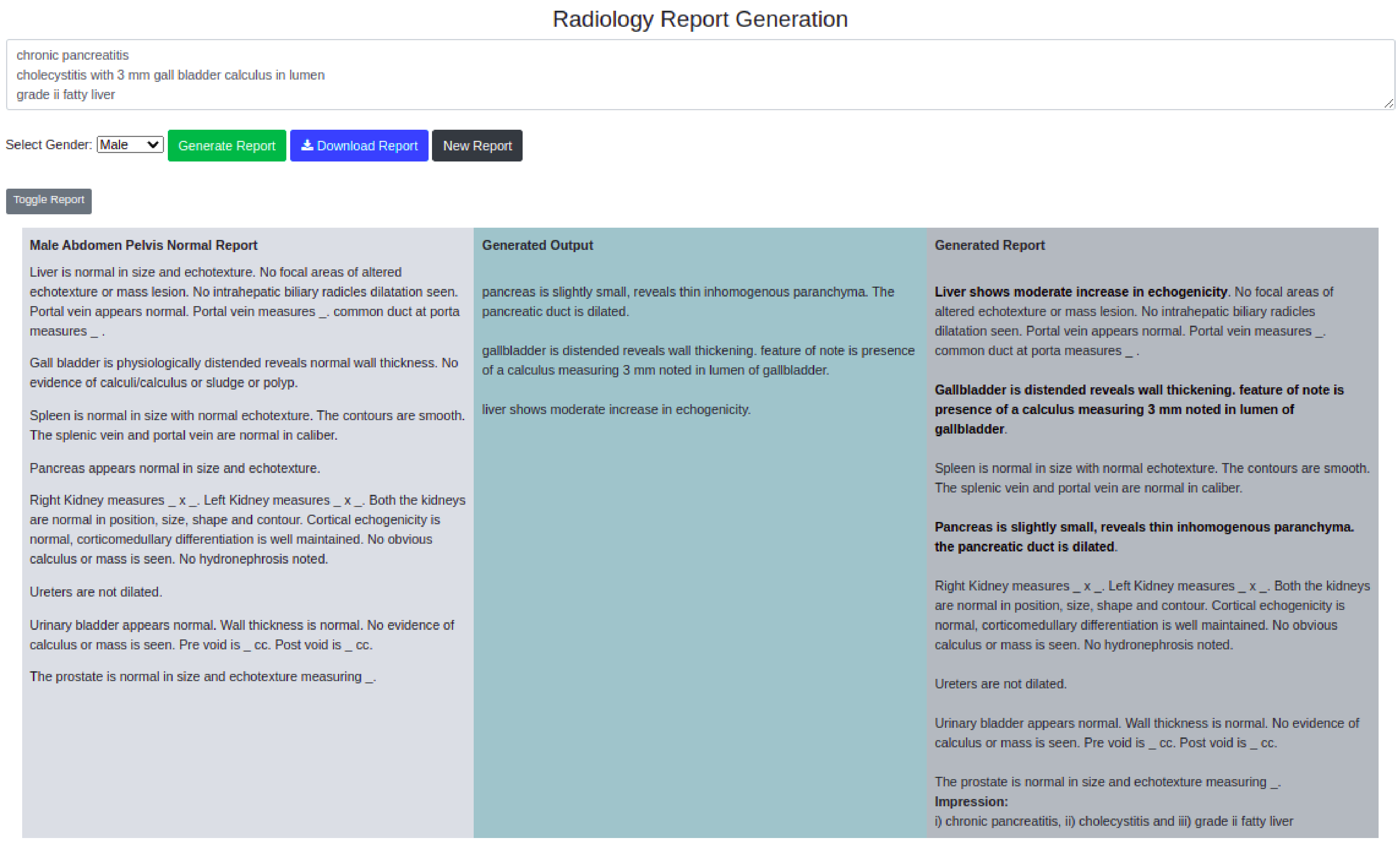
We consider following input radiologists' notes: *'Chronic pancreatitis.', 'Cholecystitis with 3 mm gallbladder calculus in lumen.'* and *'Grade ii fatty liver.'* and their corresponding generated pathological descriptions by our radiology text generator, *'Pancreas is slightly small, reveals thin inhomogenous parenchyma. the pancreatic duct is dilated.', 'Gallbladder is distended reveals wall thickening. feature of note is presence of a calculus measuring 3 mm noted in lumen of gallbladder.'* and *'Liver shows moderate increase in echogenicity.'* respectively.

- **Step 1:** We look up similar radiologists' notes to the input radiologist's notes in a parallel corpus shown in the table 6. We utilize the BLEU score to determine the match. Then, we retrieve the appropriate normal description for the matching sample. For example, for input notes, *'chronic pancreatitis'* algorithm gives matched radiologists' notes *'chronic pancreatitis'*. Similarly, for input notes, *'cholecystitis with 3 mm gallbladder calculus in lumen'* algorithm gives corresponding matched notes *'cholecystitis with gallbladder calculus in lumen.'* and for input notes *'grade ii fatty liver'* algorithm gives corresponding matched notes *'Fatty liver'*.
- **Step 2:** We find the normal sentences in normal report template which matches with normal descriptions found in step 1.
- **Step 3:** We replace matched normal sentences in the template with the corresponding generated pathological descriptions.

Figure 3 shows the system interface of radiology report generation.

10 Conclusion

We have constructed a parallel dataset of radiologists' notes and corresponding pathological descriptions. KG-BART for radiology text generation produces high-quality sentences by capturing relationships between the concepts in the input. It also considers default properties from the KG if they are missing in the input concept set to generate more logical and natural sentences. Our approach to construct grounded KGs does not add noise since it only considers entities in the hierarchical path from concept to root node and only adds neighbors with default properties. Experimental results show



chronic pancreatitis
cholecystitis with 3 mm gall bladder calculus in lumen
grade ii fatty liver

Male Abdomen Pelvis Normal Report

Liver is normal in size and echotexture. No focal areas of altered echotexture or mass lesion. No intrahepatic biliary radicles dilatation seen. Portal vein appears normal. Portal vein measures _. common duct at porta measures _ .

Gall bladder is physiologically distended reveals normal wall thickness. No evidence of calculi/calculus or sludge or polyp.

Spleen is normal in size with normal echotexture. The contours are smooth. The splenic vein and portal vein are normal in caliber.

Pancreas appears normal in size and echotexture.

Right Kidney measures _ x _. Left Kidney measures _ x _. Both the kidneys are normal in position, size, shape and contour. Cortical echogenicity is normal, corticomedullary differentiation is well maintained. No obvious calculus or mass is seen. No hydronephrosis noted.

Ureters are not dilated.

Urinary bladder appears normal. Wall thickness is normal. No evidence of calculus or mass is seen. Pre void is _ cc. Post void is _ cc.

The prostate is normal in size and echotexture measuring _.

Generated Output

pancreas is slightly small, reveals thin inhomogenous paranchyma. The pancreatic duct is dilated.

gallbladder is distended reveals wall thickening. feature of note is presence of a calculus measuring 3 mm noted in lumen of gallbladder.

liver shows moderate increase in echogenicity.

Generated Report

Liver shows moderate increase in echogenicity. No focal areas of altered echotexture or mass lesion. No intrahepatic biliary radicles dilatation seen. Portal vein appears normal. Portal vein measures _. common duct at porta measures _ .

Gallbladder is distended reveals wall thickening. feature of note is presence of a calculus measuring 3 mm noted in lumen of gallbladder.

Spleen is normal in size with normal echotexture. The contours are smooth. The splenic vein and portal vein are normal in caliber.

Pancreas is slightly small, reveals thin inhomogenous paranchyma. the pancreatic duct is dilated.

Right Kidney measures _ x _. Left Kidney measures _ x _. Both the kidneys are normal in position, size, shape and contour. Cortical echogenicity is normal, corticomedullary differentiation is well maintained. No obvious calculus or mass is seen. No hydronephrosis noted.

Ureters are not dilated.

Urinary bladder appears normal. Wall thickness is normal. No evidence of calculus or mass is seen. Pre void is _ cc. Post void is _ cc.

The prostate is normal in size and echotexture measuring _.
Impression:
i) chronic pancreatitis, ii) cholecystitis and iii) grade ii fatty liver

Figure 3: UI interface of our Radiology Report Generation system. The upper text area shows the inputs (radiologist's notes) given to the system. Drop-box gives the option to select the template for male or female. After clicking on **Generate Report** button, the system generates a pathological description using our KG-BART radiology text generator and then generates the whole report. The lower first column shows the normal report template. We display the generated pathological description in the lower mid column. The third column shows the generated report. Once the report is generated, we can download it by clicking on the **Download Report** button.

Input	bulky and retroverted uterus 1 x 5 x 3.4 mm with endometrial thickness 7 mm.
T5-large	uterus is retroverted and bulky in size 1 x 5 x 3.4 mm. myometrial reflectivity is inhomogeneous with an echogenic lesion measuring 7 mm in the mid anterior wall.
BART-large	Uterus is retroverted and bulky in size 1 x 5 & 3.4 mm. Myometrial reflectivity is inhomogeneous and shows normal echotexture. It has uniform epigastrophic echopattern.
KG-BART	uterus is retroverted and bulky in size 1 x 5 x 3.4 mm. myometrial reflectivity is inhomogeneous. Endometrial thickness is 7 mm in size.
Target	Uterus is retroverted and bulky in size measuring 1 x 5 x 3.4 mm. myometrial reflectivity is inhomogeneous and shows bulky echotexture. Endometrial thickness is 7 mm in size.
Input	gross splenomegaly, maximum span of spleen is 8.2 mm with focal calcifications noted in spleen.
T5-large	e/o spleen is grossly enlarged and normal in echotexture, maximum span of stiple is 8.2 mm. multiple calcified granulomas noted in adnexa.
BART-large	spleen is grossly enlarged and normal in echotexture, maximum span of spleen is 8.2 mm. multiple calcified granulomas noted in gb.
KG-BART	Spleen is grossly enlarged and normal in echotexture, maximum span of spleen is 8.2 mm. multiple calcified granulomas noted in spleen.
Target	Spleen is grossly enlarged and normal in echotexture, maximum span of spleen is 8.2 mm. Multiple calcified granulomas noted in spleen.

Table 5: Examples of input (radiologist's notes) and output (pathological description) generated by T5-large, BART-large and KG-BART model.

Radiologists' Notes	**Normal Description**
fatty liver	Liver is normal in size and echotexture.
acute pancreatitis	Pancreas is normal in size and echotexture.
chronic pancreatitis	Pancreas is normal in size and echotexture.
cholecystitis with gallbladder calculus in lumen	Gall bladder is physiologically distended reveals normal wall thickness. No evidence of calculi/calculus or sludge or polyp.

Table 6: Samples from the parallel corpus of radiologists' notes and normal descriptions.

that KG-BART is more capable of producing radiology text than the state-of-the-art T5-base/large and BART-base/large models. In future, we plan to apply the proposed method to generate radiology reports for CT, MRI, *etc.*

Limitations

Available medical reports used to construct the parallel dataset are biased as the abnormal findings for the liver, pancreas, kidney, gallbladder, and uterus are more weighted than organs like ovary, prostate, urinary bladder, etc. Hence, generated pathological descriptions for organs with fewer data are influenced by data with highly weighted organs.

Broader Impact

Automatic radiology report generation can augment radiologists' capabilities to enhance clinical workflows. Our work will help to avoid delays in report turnaround time and human typographical errors in the report. It will speed up the report generation process resulting in faster medical treatment. This will help faster treatment of patients thereby saving many lives.

References

Richa Arora. 2014. The training and practice of radiology in india: current trends. *Quantitative imaging in medicine and surgery*, 4(6):449.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Shivani Choudhary, Tarun Luthra, Ashima Mittal, and Rajat Singh. 2021. A survey of knowledge graph embedding and their applications. *arXiv preprint arXiv:2107.07842*.

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*.

Janice C Honeyman. 1999. Information systems integration in radiology. *Journal of Digital Imaging*, 12(1):218–222.

Kaveri Kale, Pushpak Bhattachryya, Aditya Shetty, Milind Gune, Kush Shrivastava, Rustom Lawyer, and

Spriha Biswas. 2022. Knowledge graph construction and its application in automatic radiology report generation from radiologist's dictation. *arXiv preprint arXiv:2206.0630808*.

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.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-bert: Enabling language representation with knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2901–2908.

Ye Liu, Yao Wan, Lifang He, Hao Peng, and S Yu Philip. 2021. Kg-bart: Knowledge graph-augmented bart for generative commonsense reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6418–6425.

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.

Ofir Press and Lior Wolf. 2016. Using the output embedding to improve language models. *arXiv preprint arXiv:1608.05859*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Wenhao Yu, Chenguang Zhu, Zaitang Li, Zhiting Hu, Qingyun Wang, Heng Ji, and Meng Jiang. 2022. A survey of knowledge-enhanced text generation. *ACM Computing Surveys (CSUR)*.

Yixiao Zhang, Xiaosong Wang, Ziyue Xu, Qihang Yu, Alan Yuille, and Daguang Xu. 2020. When radiology report generation meets knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12910–12917.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. Ernie: Enhanced language representation with informative entities. *arXiv preprint arXiv:1905.07129*.

A Appendix

A.1 Data Preprocessing

In the corpus, there are a lot of extra spaces and unwanted punctuation marks found. We have removed these unwanted characters from the corpus using regular expressions.

For example, *Liver is enlarged in size(16. 45cm)& normal in shape and shows raised echo reflectivity. No focal or diffuse lesion is seen. The portal and hepatic veins are normal.* In the above example, there is no space between *size, (16.45cm)* and *&*. Also, there is no space between . and *No* and therefore sentence tokenization is challenging. *Liver is enlarged in size (16.45 cm) & normal in shape and shows raised echo reflectivity. No focal or diffuse lesion is seen. The portal and hepatic veins are normal.* The text is then further divided into sentences.

A.1.1 Spelling Correction

In corpus, there are a lot of spelling mistakes also. To correct the spellings we have used the SymSpell library.

Single Word Spelling Correction We have created unigram and bigram dictionaries for corpus text.
Unigram Dictionary: Dictionary of unique correct spelling words, and the frequency count for each word.
Bigram Dictionary: Dictionary of the unique correct spelling of a pair of words, and the frequency count for each pair.

Levenshtein algorithm is used to compute edit distance metric between two strings. Edit distance algorithm finds the correct suggestion for words in input text with words in unigram dictionary.

For example, *enlaregd, billiary, radicals* are the incorrect words found in the corpus. In dictionary enlarged, biliary, radicals these correct words are present. Edit distance algorithm suggests enlarged word for *enlaregd*. Similarly *biliary* for *billiary* and *radicles* for *radicals*.

Multi-word Spelling Correction

- We remove mistakenly inserted spaces within a correct word

Input: *Liver is normal in size and reveals diffuse hypo attenuation*
Output: *Liver is normal in size and reveals diffuse hypoattenuation*

- We add mistakenly omitted spaces between two correct words
  Input: *Liver appears normal in size and reveals mild generalized increasedparenchymal reflectivity.*
  Output: *Liver appears normal in size and reveals mild generalized increased parenchymal reflectivity.*

Named Entity Recognition for Code-Mixed Kannada-English Social Media Data

Poojitha Nandigam, Appidi Abhinav Reddy, Manish Shrivastava
Language Technologies Research Centre (LTRC)
International Institute of Information Technology, Hyderabad, India
poojitha.nandigam@research.iiit.ac.in, abhinav.appidi@research.iiit.ac.in,
m.shrivastava@iiit.ac.in

Abstract

Named Entity Recognition (NER) is a critical task in the field of Natural Language Processing (NLP) and is also a sub-task of Information Extraction. There has been a significant amount of work done in entity extraction and Named Entity Recognition for resource-rich languages. Entity extraction from code-mixed social media data like tweets from twitter complicates the problem due to its unstructured, informal, and incomplete information available in tweets. Here, we present work on NER in Kannada-English code-mixed social media corpus with corresponding named entity tags referring to Organisation (Org), Person (Pers), and Location (Loc). We experimented with machine learning classification models like Conditional Random Fields (CRF), Bi-LSTM, and Bi-LSTM-CRF models on our corpus.

1 Introduction

India has twenty-three significant languages with over seven hundred and twenty dialects. Kannada is one of the four major Dravidian languages and it is one of the top 30 most spoken languages of the world, with its own independent script and over fifty million speakers. The majority of people are multilingual and tend to mix words from different languages in speech and written text. This method of interchanging languages involves complex grammar and is commonly addressed by terms ‘Code-switching’ and ‘Code-mixing’ as described by Lipski (1978).

Code-mixing refers to the use of words, phrases, clauses or morphemes from different languages in the same sentence. Code-switching refers to the use of words or phrases or clauses from different languages within the same speech context. We can understand the difference between code-switching and code-mixing from the positions of altered elements. Code-mixing refers to the intra-sentential modification of codes, whereas code-switching refers to the inter-sentential modification of codes.

1.1 Characteristic of Code-Mixed Kannada-English Data

As explained above mixing happens at word level, phrase level, and morphological level too. Following are few more examples :

1. **Word level:** A complete word from English language is taken into Kannada language. This is language mixing occuring at word level. An example: ‘Ee thara branch ideya’ which means ‘Is there a branch like this?’. Here ‘branch’ is an English word which got assimilated into Kannada.

2. **Phrase level:** This is a completely code-mixed sentence, that follows the structure of Kannada with English words embedded in it. One example is ‘Kelsa bittu pitch reporter aagu olle future ide!’ which means ‘Leave your work and become pitch reporter, you have great future in that!’, this follows the structure of Kannada with English words embedded in it. This is a completely code-mixed sentence.

3. **Morphological level:** The words that are borrowed from English language inflect Kannada suffixes that marks case or number. The word ‘cinemagalu’ in Kannada, the root word ‘cinema’ is borrowed from English and ‘galu’ is a Kannada morphene that marks plurality. Similarly ‘caru’ becomes ‘car’, this is nativization.

4. **Syntactic level:** All the examples above are instances of intra-sentential mixing. Here we discuss about intra-sentential and inter-sentential mixing. There are occurrences in Kannada-English CM data where inter-sentential mixing takes place. One such example is ‘Born and brought up in bengaluru,

Yaako nange mysoor thumba ista, mysoor alli kelsa sikdre ready to shift.'

We observe code-switching and code-mixing frequently on social media platforms like Facebook and Twitter. Here, we work only on the code-mixing aspect observed on Twitter data between Kannada and English languages.

Understanding the code-mixed Kannada-English complicates the problem due to its unstructured, informal, and incomplete information available in the data. Following are the challenges associated when dealing with them.

- **Ambiguous words**: Same word can have a different meaning in multiple languages. Like the word 'Bali' in English, which is a place in Indonesia, also used in Kannada with different meaning here as 'Near'.

- **Variable Lexical Representations**: Some users prefer to use their own romanised form of native word. For example 'hogilla' is a Kannada word and it can be written as 'hogila', 'hgilla', 'hogillla' etc.

- **Word-level Code-mixing**: This is similar to language mixing at word-level. For example in the word 'Kanglish', its a fusion of two words Kannada and English at word level.

- **Reduplication**: This is common in Indian languages. People tend to use a second word which does not have a meaning on its own but with the first word it becomes a multi word expression. For example 'postu geestu', 'desha gesha', 'man ban'. The first words in these examples are English which are followed by reduplicated words.

Here are some instances from a corpus of Kannada-English generated from Twitter data and also transliterated in English.

T1 : *"Haha ashtu idea illade gowdru bengaluru north bittu tumukur hogilla"*
Translation: *"Haha without having much idea gowda left bengaluru north and went to tumukur "*

T2: *"Eshwarappa avarey neevu petrol bunk ge hogilla ansuthe. me nimmannu karkondu hogthini"*
Translation: "Eshwarappa, it looks like you did not go to the petrol bunk. I will take you there."

2 Background and Related Work

There has been a plethora of research done on Named Entity Recognition (NER) from the early 2000s (Finkel et al., 2005). However, most of this is in resource-rich languages. The FIRE2 (Forum for Information Retrieval and Extraction) tasks have shed light on NER in Indian languages. Now, code-mixing has found its application in various areas after FIRE2, such as Query Labeling (Bhargava et al., 2015), Sentiment Analysis (Bhargava et al., 2016), Question Classification.

BR and Ramakanth Kumar (2012) has done the work on the Kannada POS tagger with probabilistic classifiers. Similar work has been done by Todi et al. (2018) in the Kannada POS tagger using machine learning models. Amarappa and Sathyanarayana (2013) worked on NER and classification in the Kannada language. Lakshmi and Shambhavi (2017) presented an automatic identification system for code-mixed Kannada-English Social media text. Shalini et al. (2018) worked on sentiment analysis for Code-Mixed Kannada-English Social Media Text. To the best of our knowledge, the corpus we created is the first Kannada-English code-mixed corpus with named entity tags.

3 Corpus and Annotation

This corpus consists of Kannada-English code mixed tweets gathered from twitter. The tweets were collected using twintproject[1]-an opensource twitter intelligence tool. The tweets are from the past 6 years based on various topics such as movies, sports, celebrities, politics, trending hashtags, social events.

We have retrieved a total of over 317,000 tweets using the twintproject, and after extensive cleaning and pre-processing, we were left with 6530 Kannada-English code mix tweets.

The pre-processing consists of the following steps.

- Removing noisy, useless tweets, i.e., tweets containing only URLs and hashtags.

- Tweets which were written in only Kannada, or only English were removed too.

- Tweets which contain linguistic units from both English and Kannada and having a minimum of five words are only considered, this

[1]https://github.com/twintproject/twint

way, we make sure the tweets adhere to the Kannada-English code mix standard.

- Tokenisation of tweets is done using Tweet Tokenizer.

The corpus will be made available for public use as soon as possible. The following explains the mapping of the tokens with their respective tags.

3.1 Annotation: Named Entity Tagging

We used three Named Entities (NE) tags “Person," “Location," and “Organisation" to tag the corpus. Two people manually did the annotations of the data for Named Entity tags. The annotators have a linguistic background, and are proficient in both Kannada and English. Each of three tags (“Person," “Location" and “Organisation") is divided into Bg-tag (Beginner tag) and It-tag (Intermediate tag) according to the BIO standard thus we have a total of six tags and an ‘Other’ tag to indicate it does not belong to any of the six tags. The Bg-tag is used to tag the beginning word of a Named Entity, whereas It-tag is used to tag a Named Entity, which is split into multiple continuous words. It-tag is assigned to the words which follow the words with a Bg-tag. The explanation of six tags is below.

The ‘Pers’ tag refers to the ‘Person’ entity, which is the name of the person, twitter handles and nicknames of people. The ‘Bg-Pers’ tag is given to the beginning word of a person’s name, and the ‘It-Pers’ tag follows ‘Bg-pers’ tag, if the person’s name is split into multiple continuous words.

The ‘Loc’ tag refers to the ‘Location’ entity, which is the name of the place like Bangalore, Hyderabad, India and others. The ‘Bg-Loc’ tag is assigned to the beginning word of the location name, and the ‘It-Loc’ tag follows ‘Bg-Loc’ tag, if the location name is split into multiple continuous words.

The ‘Org’ tag refers to the ‘Organisation’ entity, which is the name of the organization such as BJP, KFI, INC, Facebook, RBI, and others. The ‘Bg-Org’ tag is assigned to the beginning word of the organization name, and the ‘It-Org’ tag follows ‘Bg-Org’ tag, if the organization name is split into multiple continuous words. Following is an example that shows the application of principles described above.

T3 : *“Haha/other ashtu/other idea/other illade/other gowdru/Bg-Per bengaluru/Bg-Loc north/It-Loc bittu/other tumukur/Bg-Loc hogilla/other”*
Translation: “Haha without having much idea gowda left bengaluru north and went to tumukur."

T4 : *“@vs20012000/Bg-Per illa/other hogilla.../other Harish/Bg-Per ex/other Deputy/Bg-Org Mayor/It-Org organised/other volleyball/other tourney/other ge/other swalpa/other kelasa/other madidde...now/other in/other Bombay./Bg-Loc”*
Translation: “No, did not go.. i did a little bit of work for the volleyball tournament organized by Harish, ex-Deputy Mayor. Now i am in Bombay.."

Tag	Token Count	Cohen Kappa
Bg-Loc	1457	0.89
Bg-Org	3178	0.94
Bg-Pers	5899	0.88
It-Loc	188	0.84
It-Org	505	0.89
It-Pers	358	0.82
Total NE tokens	11585	

Table 1: Tags and their Count in Corpus and IAA.

3.2 Inter Annotator Agreement

The annotations of the data for Named Entity tags were manually done by two people with linguistic backgrounds, both proficient in Kannada and English. The quality of the annotation is validated using the inter-annotator agreement (IAA) between two annotation sets of 6,530 tweets and 152,987 tokens using Cohen’s Kappa coefficient (Hallgren, 2012) (refer Table 1 for Score). The agreement is significantly high. The agreement between the ‘Organisation’ tokens is high while that of ‘Location’ and ‘Person’ tokens is comparatively low due to unclear context and presence of an uncommon or confusing person and location names.

4 Corpus Statistics

We have collected more than 317,000 of tweets from Twitter using TwintProject. After extensive cleaning, we were left with 6,530 Kannada-English code mixed tweets, as part of annotation using six named entity tags along with ‘Other’ tag we tagged 152,987 tokens. We made sure that all the words in the corpus are in Roman script. We used hashtags related to politics, sports, social events and recent trends etc., in collecting the corpus.

5 Experiments

We present all experiments using a combination of features and systems. To understand the effect of different parameters and features of the model, we performed several experiments. Experiments were performed using some set of features at once and all at a time simultaneously changing the parameters of the model, like regularization parameters and algorithms of optimization like 'L2 regularization', 'Average Perceptron' and 'Passive Aggressive' for CRF, optimization algorithms and loss functions in LSTM. We used five-fold cross-validation for CRF and three-fold for other experiments in order to validate our classification models. We used 'scikit-learn,' 'Tensorflow,' and 'Keras' libraries for the implementation of the above algorithms.

5.1 Conditional Random Field (CRF)

A Conditional Random Field (CRF) is an undirected probabilistic graphical model that is used for modeling sequential data. It is a model for predicting the most likely sequence of labels that correspond to a sequence of inputs. It has applications in POS tagging, NER, among others. It is a supervised learning method and most often used for structured prediction tasks. When it comes to NER, it has been proven to be better than the tree-based models. Whereas a discrete classifier predicts a label for a single sample without considering "neighboring" samples, a linear chain CRF can take context into account and predicts sequences of labels for sequences of input samples, which is popular in natural language processing.

5.2 LSTM

As our corpus is in sequential text data format, we use Bi-LSTM (combination of two LSTMs — where one runs forward, and one runs backward), which works best to tackle the NER problem as the context covers both past and future labels in a sequence because standard LSTM makes use of only past information in a sequence of text and not the future. Plain LSTM cells in a feedforward network which help us in getting better results by capturing the previous context while Bi-LSTMs also consider the opposite direction. Bi-LSTM considers a sequence of both tokens that are before and after a token of interest. Bi-LSTM network creates a context for each token in the text, which depends on both its past and future.

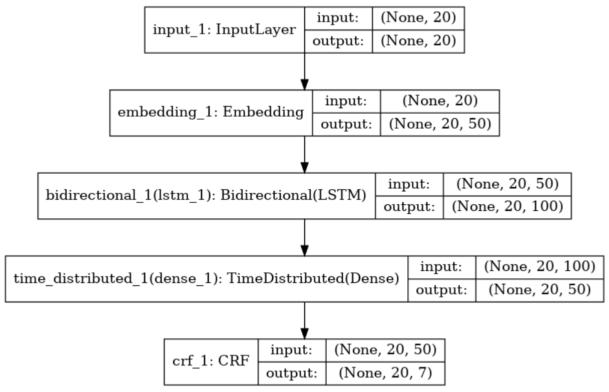


Figure 1: BiLSTM-CRF model architecture

5.3 LSTM-CRF

The Bi-LSTM-CRF is a combination of bidirectional LSTM and CRF (Huang et al., 2015; Lample et al., 2016). The Bi-LSTM model can be combined with CRF to enhance recognition accuracy. This combined model of Bi-LSTM-CRF inherits the ability to learn past and future context features from the Bi-LSTM model and use sentence-level tags to predict possible tags using the CRF layer. We processed the data in batches and used seven epochs.

5.4 Features

The features to our machine learning models consist of characters, lexical and word-level features such as char N-Grams of size 2 and 3 in order to capture the information from suffixes, emojis, mentions in social media like '#,' '@,' punctuation, numbers, numbers in the string. Features from adjacent tokens are used as contextual features.

1. **Character N-Grams:** N-gram is a contiguous sequence of n items from a given sample of text or speech, here the items are characters. Character N-Grams are language-independent (Majumder et al., 2002) and have proven to be efficient in the task of text classification. They are helpful when the text suffers from problems such as misspellings. (Cavnar et al., 1994; Huffman, 1995; Lodhi et al., 2002). Group of chars can help in capturing the semantic information and especially helpful in cases like code mixed language where there is free use of words, which vary significantly from the standard Kannada-English words.

2. **Word N-Grams:** Bag of words has been a staple in NER tasks for languages other than

English (Jahangir et al., 2012). Thus, we use word N-Grams, where we use adjacent words as a feature vector to train our model. These are also called contextual features.

3. **Capitalization:** In social media, people tend to use capital letters to refer to the names of locations, persons, and organizations; at times, they write the entire name in capitals (Von Däniken and Cieliebak, 2017) to give particular importance or to denote aggression. This gives rise to a couple of binary features. One feature is to indicate if the beginning letter of a word is capitalized, and the other is to indicate if the entire word is capitalized.

4. **Mentions and Hashtags:** In social media organizations, like Twitter and Facebook, people use '@' mentions to refer to persons or organizations, they use '#' hashtags in order to make something notable or to make a topic trending. Thus the presence of these two gives a reasonable probability for the word being a named entity.

5. **Numbers in String:** In social media, we see people using alphanumeric characters, generally to save the typing effort, shorten the message length or to showcase their style. When observed in our corpus, words containing alphanumeric are generally not named entities. Thus the presence of alphanumeric in words helps us in identifying negative samples.

6. **Common Symbols:** It is observed that currency symbols, brackets like '(,' '[,' etc. And other symbols are followed by numeric or some mention, not of much importance. Hence, the presence of these symbols is a good indicator of the words before or after them for not being a named entity.

6 Results and Discussion

Table 2 shows CRF results with 'l2-sgd' (Stochastic Gradient Descent with L2 regularization) algorithm for 100 iterations. The c2 value in the CRF model refers to the 'L2 regression,'. Experiments using the algorithms 'pa' (Passive-Aggressive) and 'ap' (Averaged Perceptron) resulted in similar F1-scores of 0.95.

Results for CRF without 'Other' tag are shown in Table 3 which resulted in F1-score of 0.54. We can observe from the results that the feature functions specified are able to capture information related to named entities in the CRF model. The table for feature specific results for the CRF model where results are calculated excluding the 'Other' tag shown in Table 4.

In the experiments with Bi-LSTM, we experimented with the optimizer, activation functions,and number of epochs. After several experiments, the best result we came through was using 'softmax' as activation function, 'rmsprop' as an optimizer,

Tag	Precision	Recall	F1-score
Bg-Loc	0.83	0.48	0.61
Bg-Org	0.83	0.52	0.64
Bg-Pers	0.85	0.55	0.67
It-Loc	0.68	0.27	0.38
It-Org	0.52	0.22	0.31
It-Pers	0.58	0.27	0.37
OTHER	0.96	0.99	0.98
weighted avg	0.95	0.96	0.95

Table 2: CRF Model with 'c2=0.1' and 'l2sgd' algo.

Tag	Precision	Recall	F1-score
Bg-Loc	0.73	0.28	0.40
Bg-Org	0.77	0.32	0.45
Bg-Pers	0.76	0.61	0.67
It-Loc	0.33	0.01	0.01
It-Org	0.15	0.03	0.05
It-Pers	0.51	0.06	0.10
weighted avg	0.72	0.44	0.54

Table 3: CRF Model without 'Other' tag, 'c2=0.1' and 'l2sgd' algo.

Feature	Precision	Recall	F1-score
Char N-Grams	0.68	0.38	0.49
Word N-Grams	0.55	0.08	0.14
Capitalization	0.85	0.44	0.58
Mentions, Hashtags	0.72	0.26	0.38
Numbers in String	0.01	0.01	0.01
Common Symbols	0.02	0.02	0.02

Table 4: Feature Specific Results for CRF.

Tag	Precision	Recall	F1-score
Bg-Loc	0.72	0.32	0.44
Bg-Org	0.69	0.55	0.61
Bg-Pers	0.74	0.72	0.73
It-Loc	0.60	0.06	0.11
It-Org	0.39	0.09	0.15
It-Pers	0.58	0.23	0.33
OTHER	0.98	0.99	0.99

Table 5: Bi-LSTM model with optimizer = 'rmsprop' and has a weighted f1-score of 0.96.

Tag	Precision	Recall	F1-score
Bg-Loc	0.76	0.41	0.54
Bg-Org	0.74	0.49	0.59
Bg-Pers	0.85	0.44	0.58
It-Loc	0.03	0.02	0.02
It-Org	0.24	0.06	0.10
It-Pers	0.26	0.03	0.05

Table 6: Bi-LSTM model without 'Other' tag, optimizer = 'rmsprop' and has a weighted f1-score of 0.54.

and 'categorical cross-entropy' as our loss function. Table 5 shows the results of BiLSTM on our corpus using seven epochs, and random initialization of embedding vectors. The F1-score is 0.96.
Results for same experiment without including 'Other' tag are shown in Table 6.

In experiments with the Bi-LSTM-CRF model, after several trials, we got the best results with 'softmax' as activation function, 'rmsprop' as an optimizer, and 'crf-loss' as our loss function. Table 7 shows the results of Bi-LSTM-CRF on our corpus using seven epochs, and random initialization of embedding vectors. The F1-score is 0.96.
Results for same experiment without including 'Other' tag are shown in Table 8. Figure 1 shows the Bi-LSTM-CRF model architecture. The training, validation and testing are 70%, 10% and 20% of the total data respectively.

7 Conclusion and Future Work

Our contributions are as follows:

1. Presented an annotated code-mixed Kannada-English corpus for NER, which is, to the best of our knowledge is the first corpus. The corpus will be published online soon.

2. We have experimented with the machine learning models Conditional Random Fields (CRF), LSTM, and LSTM-CRF on our data, the F1-score for which is 0.95, 0.96, and 0.96 respectively, which is looking good considering the amount of research done in this new domain.

3. We are introducing and addressing named entity recognition of Kannada-English code-mixed data as a research problem.

Tag	Precision	Recall	F1-score
Bg-Loc	0.65	0.40	0.49
Bg-Org	0.61	0.65	0.63
Bg-Pers	0.57	0.80	0.66
It-Loc	0.50	0.22	0.31
It-Org	0.30	0.25	0.27
It-Pers	0.54	0.41	0.47
OTHER	0.99	0.98	0.98

Table 7: Bi-LSTM-CRF model with optimizer = 'rmsprop' and has a weighted f1-score of 0.96.

Tag	Precision	Recall	F1-score
Bg-Loc	0.75	0.41	0.53
Bg-Org	0.72	0.50	0.59
Bg-Pers	0.85	0.44	0.58
It-Loc	0.25	0.01	0.02
It-Org	0.32	0.16	0.21
It-Pers	0.28	0.04	0.08

Table 8: Bi-LSTM-CRF model without 'Other' tag, optimizer = 'rmsprop' and has a weighted f1-score of 0.55.

Word	Truth	Predicted
amrita	Bg-Pers	Bg-Pers
went	OTHER	OTHER
to	OTHER	OTHER
bangalore	Bg-Loc	Bg-Loc
rama	Bg-Pers	Bg-Pers
na	OTHER	OTHER
imax	Bg-Org	Bg-Org
hattira	OTHER	OTHER
beti	OTHER	Bg-Pers
agalu	OTHER	OTHER

Table 9: An Example Prediction of our CRF Model.

For future work, the corpus can be enriched by also giving the respective POS tags for each token. The size of the corpus can be increased with more NE tags. The problem can be adapted for NER identification in code-mixed data containing more than two languages from multilingual societies.

References

S Amarappa and SV Sathyanarayana. 2013. Named entity recognition and classification in kannada language. *International Journal of Electronics and Computer Science Engineering*, 2(1):281–289.

Rupal Bhargava, Yashvardhan Sharma, and Shubham Sharma. 2016. Sentiment analysis for mixed script indic sentences. In *2016 ICACCI*, pages 524–529. IEEE.

Rupal Bhargava, Yashvardhan Sharma, Shubham Sharma, and Abhinav Baid. 2015. Query labelling for indic languages using a hybrid approach. In *FIRE Workshops*, pages 40–42.

Shambhavi BR and P Ramakanth Kumar. 2012. Kannada part-of-speech tagging with probabilistic classifiers. *international journal of computer applications*, 48(17):26–30.

William B Cavnar, John M Trenkle, et al. 1994. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*, volume 161175. Citeseer.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics.

Kevin A Hallgren. 2012. Computing inter-rater reliability for observational data: an overview and tutorial. *Tutorials in quantitative methods for psychology*, 8(1):23.

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

Stephen Huffman. 1995. Acquaintance: Language-independent document categorization by N-grams. Technical report, DEPARTMENT OF DEFENSE FORT GEORGE G MEADE MD.

Faryal Jahangir, Waqas Anwar, Usama Ijaz Bajwa, and Xuan Wang. 2012. N-gram and gazetteer list based named entity recognition for Urdu: A scarce resourced language. In *Proceedings of the 10th Workshop on Asian Language Resources*, pages 95–104.

BS Sowmya Lakshmi and BR Shambhavi. 2017. An automatic language identification system for code-mixed english-kannada social media text. In *2017 2nd International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS)*, pages 1–5. IEEE.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

John Lipski. 1978. Code-switching and the problem of bilingual competence. *Aspects of bilingualism*, 250:264.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2(Feb):419–444.

P Majumder, M Mitra, and BB Chaudhuri. 2002. N-gram: a language independent approach to IR and NLP. In *International conference on universal knowledge and language*.

K Shalini, HB Barathi Ganesh, M Anand Kumar, and KP Soman. 2018. Sentiment analysis for code-mixed indian social media text with distributed representation. In *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1126–1131. IEEE.

Ketan Kumar Todi, Pruthwik Mishra, and Dipti Misra Sharma. 2018. Building a kannada pos tagger using machine learning and neural network models. *arXiv preprint arXiv:1808.03175*.

Pius Von Däniken and Mark Cieliebak. 2017. Transfer learning and sentence level features for named entity recognition on tweets. In *3rd Workshop on Noisy User-generated Text (W-NUT), Copenhagen, 7 September 2017*, volume 3, pages 166–171. ACL.

A Example Appendix

This is an appendix.

PAR: Persona Aware Response in Conversational Systems

Abhijit A Nargund
Samsung R & D Institute
Bangalore, India
abhijit.an@samsung.com

Sandeep Kumar Pandey
Samsung R & D Institute
Bangalore, India
sandeep.p@samsung.com

Jina Ham
Samsung Research
Korea
j.ham@samsung.com

Abstract

To make the Human Computer Interaction more user friendly and persona aligned, detection of user persona is of utmost significance. Towards achieving this objective, we describe a novel approach to select the persona of a user from pre-determine list of personas and utilize it to generate personalized responses. This is achieved in two steps. Firstly, closest matching persona is detected from a set of pre-determined persona for the user. The second step involves the use of a fine-tuned natural language generation (NLG) model to generate persona compliant responses. Through experiments, we demonstrate that the proposed architecture generates better responses than current approaches by using a detected persona. Experimental evaluation on the PersonaChat dataset has demonstrated notable performance in terms of perplexity and F1-score.

1 Introduction

Since the dawn of deep learning era and advancements in Natural Language Processing (NLP), researchers have tried to make the Human-Computer Interaction (HCI) more natural and smooth (Deng and Ren, 2021). Several features are incorporated in a conversation based system to induce human-like response and give users a pleasant experience such as - sentiment analysis of user replies (Lee et al., 2018), emotion recognition (Lee et al., 2020) etc. However, there is still a gap left between the desired output and the approaches tried. A user's conversation with a bot highly depends on the general nature and likings of the user, more commonly referred to as the persona.

Many recent studies have explored persona identification from user utterances (Zhang et al., 2018; Bahdanau et al., 2014; Song et al., 2019; Natarajan and Nargund, 2021). A dataset of Persona enhanced chat between two users based on a given persona profile, known as PersonaChat was introduced in (Zhang et al., 2018). Experiments performed using several generative and ranking baseline models established the need of including persona information to enhance conversations. The study in (Gu et al., 2021) have proposed a modified dataset PMPC on PersonaChat Dataset and explored Utterance-to-Profile Matching Network for the task of many-to-many matching. Moreover, in (Liu et al., 2020), a trasmitter-reciever architecture is proposed, referred to as $\mathcal{P}^2$ bot which models the mutual persona perception using a transformer with reinforcement learning to fine tune the conversations during training. Also, to alleviate the challenges associated with data in persona based dialog generation, the study in (Cao et al., 2022) introduced data manipulation techniques such as distillation, diversification and curriculum. The architecture considered were baseline GPT2 and Transormer encoder-decoder stack.

Despite several attempts by researchers, the generation of persona complaint dialogue is very challenging. Firstly, automated metrics such as BLEU, ROGUE etc does not consider semantic similarity and will present poor scores if there is no overlap between the generated response and the gold response. Also, the response generated can be persona complaint but may not answer the query of the user. As such, we propose a novel approach to tackle the persona based dialog generation problem. Firstly, we select the matched persona corresponding to the user input from the set of pre-determined personas of the user, by using similarity matching of the embeddings. Then, we generate per-

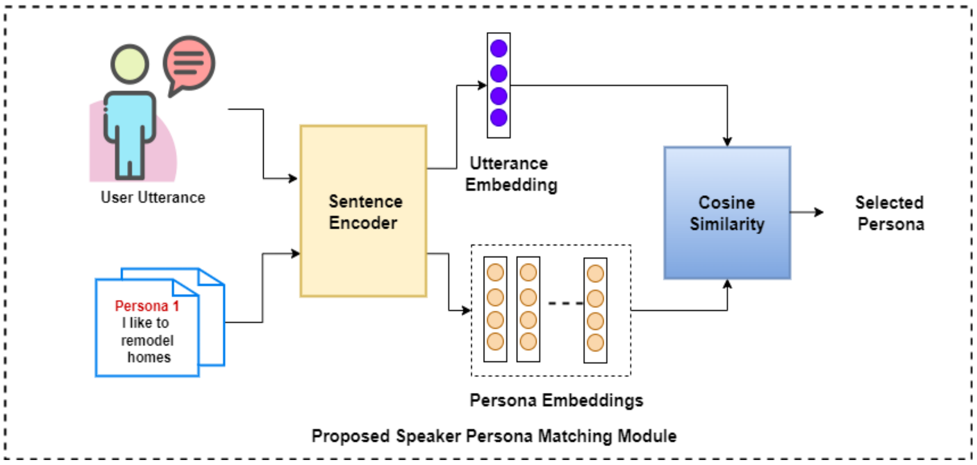


Figure 1: Proposed methodology to extract user utterance and persona embeddings and compare using cosine similarity.

sona complaint response by incorporating this additional persona information with state-of-the-art Semantically Controlled Generative Pre Training (SC-GPT)(Peng et al., 2020) model. Performance on the validation set of the PersonaChat dataset yeilds results comparable to other standard baselines.

The rest of the paper is organized as follows. Section 2 describes the methodology for persona classification and response generation. Section 3 details the experiments performed with dataset information and hypermaters used. Section 4 presents the results along with sample generated responses. Section 5 concludes the paper.

2 Methodology

This section describes the methodology used for persona selection, given a user utterance and predetermined list of persona and subsequent Natural Language Generation (NLG) module which is responsible for the generation of persona complaint bot response.

2.1 Persona Identification

For the task of choosing the most relevant persona characteristics out of the n persona sentences for the user, we employ similarity matching of sentence embeddings as shown in Figure 1. Given a user utterance X and a set of persona sentences characterizing the user choices $Z = \{z_i\}$ where $i = \{1, 2, \cdots, n\}$, the first step involves encoding the dialogue turn or the sentence X into sentence embedding X_{embed}. For this, we utilized techniques such as sentence Bert (Reimers and Gurevych, 2019), RoBERTa (Liu et al., 2019) and Universal Sentence Encoder(USE) (Yang et al., 2020), out of which USE was chosen based on better performance. In a similar fashion, embeddings from the persona sentences are calculated. Mathematically,

$$\mathcal{X}_{embed} = USE(\mathcal{X}) \quad (1)$$
$$\mathcal{Z}_{embed} = USE(z_i) \quad (2)$$

where USE denotes the pre-trained Universal Sentence Encoder, utilized to generate the latent space embeddings.

The similarity score between $\mathcal{X}$ and the set of persona embeddings $\mathcal{Z}$ is calculated using *cosine* similarity as -

$$s_i = \frac{\mathcal{X}\dot{z}_i}{\|X\|\,\|z_i\|} \quad (3)$$

where $z_i \in \mathcal{Z}$ and $\|.\|$ represents the length of the vector. The matched persona is selected by using $argmax$ over the set of scores, represented by -

$$S = \{s_i\} \quad (4)$$
$$P_{index} = argmax(S) \quad (5)$$

This matched persona is provided as a semantic input to the Natural Language Generation (NLG) module along with the gold response to find out the correlation with the candidate persona and producing persona complaint dialog response.

2.2 Response Generation

For Natural Language Generation, we have experimented with baseline GPT (Radford et al., 2018)

User utterance	Matched Persona	Generated Response
hello what are doing today	i'm a stunt double as my second job	i'm taking a break from shooting to relax
i like that ! i go to preschool	my favorite season is winter	winter is my favorite season. i like to go to the stores during the day.
hello , how are you tonight ? do you have pink and blue hair ?	i volunteer at a soup kitchen .	i'm doing great. just got back from soup kitchen helping people.
oh wow . i hope i live as long as you !	i am very strong for my age .	i hope so. i'm just over 50 now.
i just got done watching a horror movie	i read twenty books a year	scary movies are good. i read twenty books a year

Table 1: Sample responses generated on the validation set along with gold response and matched persona from user utterance.

Partition	# of Dialog	# of Persona
Train	8939	955
Validation	1000	100

Table 2: Details of the dataset- partionwise mentioning the number of complete dialog acts and possible personas.

as well as SC-GPT. To generate responses which are persona complaint, we have utilized Semantically Controlled - GPT (SC-GPT) (Peng et al., 2020) based on better perfromance compared to plain GPT. The choice of SC-GPT is motivated by its ability to produce more naturalisitc and Semantically controllable dialogue based on the input semantic form. It is pre-trained on a huge collection of plain text, dialogue-act labelled utterances as well as domain specific fine-tuning on limited data.

During training, given a user utterance, for e.g. *hi , how are you doing ? i'm getting ready to do some cheetah chasing to stay in shape .* and its corresponding classified persona - *i like to go hunting* , the input to the SC-GPT is of the form -

request(question=hi , how are you doing ? i'm getting ready to do some cheetah chasing to stay in shape.) @infrom(preference=i like to go hunting) & you must be very fast , hunting is one of my favorite hobbies

In this example, the task of SC-GPT will be to utilize the selected persona and the user utterance to generate a response which is persona complaint as well as contextually similar to the user input utterance. This is achieved by using the gold response- *you must be very fast , hunting is one of my favorite hobbies* as supervision while training.

During the inference, the gold response is omitted from the input form. The SC-GPT generates multiple responses matching with the selected persona, ranked according to the probability, and the one with the highest probability is selected.

3 Experimental Evaluation

In this section, we discuss the dataset details along with hyper-parameter tuning for the SC-GPT model for persona-based dialog generation.

3.1 Dataset Description

For the task of generating persona complaint responses, we utilize PersonaChat dataset (Zhang et al., 2018). It contains crowd-sourced multi-turn dialogues based on pre-defined personas. Table 2 shows the available number of complete dialog acts based on personas for the train and validation partitions. Evaluation for the fine-tuned model is performed using the validation set. Also, the model is fine-tuned using the entire PersonaChat train partition having approximately 17000 question-persona pair. Also, to asses low-resource scenario, we randomly sampled 1000 such pairs from the train partition to fine-tune the base SC-GPT model.

3.2 Hyperparameter Tuning

For the persona classification module, the size of sentence encodings generated is 768. The number of persona sentences considered per question for classification is 4, as in PersonaChat dataset. For the SC-GPT based fine-tuning, we first generated *(question, persona, response)* tuples for the entire training partition of the PersonaChat dataset.Training is performed in two scenarios - utilizing the entire train partition and randomly sampled 1000 tuples. For the random sampling based fine-tuning, we selected 1000 tuples out of the 17,000 generated tuples randomly. This is repeated five times and the result presented is the average over the five random folds. The model is fine-tuned for 10, 20 and 30 epochs out of which the best results were obtained at 10 epochs. The *temperature* parameter, which controls the degree of strictness of the results generated, is set at 0.7, empirically.

Method	F1 score
Seq2Seq + Attention (Bahdanau et al., 2014)	16.18
KV Profile Memory (Zhang et al., 2018)	11.9
TransferTransfo (Wolf et al., 2019)	19.09
P^2 Bot (Liu et al., 2020)	19.77
Proposed	**27.89**

Table 3: Comparison of the proposed persona complaint dialog generattion methodology with the recent state-of-the-art baseline.

Experiment	F1 score	Perplexity
Full dataset + 1 persona + 10 epochs + all past context	23.12	51.2
Full dataset + 1 persona + 20 epochs + all past context	23.03	141.59
Full dataset + 4 persona + 10 epochs + all past context	14.27	58.4
Random 1000 samples + 1 persona + 1 past context	**27.98**	**4.77**

Table 4: Ablation study for SC-GPT based persona-aware response generation based on 1 persona and 4 persona sentences.

3.3 Results

We evaluated the proposed methodology on the validation set of the PersonaChat dataset. The metric used for the evaluation of the generated response are Perplexity and F1-score. Perplexity measures the negative log likelihood of the correct sequence output by the model, lower values indicating better performance. F1 score is the harmonic mean of word-level precision and recall. Table 1 shows some of the responses generated given the persona and the user utterance. As can be observed, the generated responses are highly persona complaint. However, the gold response may vary from the generated response as the conversation is natural. Moreover, our proposed methodology achieves an F1 score of 27.89 and perplexity of 4.77 on the response generation task.

Table 3 shows the comparison of the proposed approach with the state-of-the-art methods on PersonaChat Dataset. Our proposed approach outperforms the baseline $\mathcal{P}^2$ bot by a margin of 8.12, thus highlighting the capability of the proposed approach in generating persona aware responses.

3.4 Ablation Study

We performed ablation study to assess the effectiveness of the persona selection module in improving the persona complaint response generation. To this end, we investigated three aspects - the use of full training partition vs. 1000 randomly selected training tuples, increased training epochs and the use of full persona profile(4 persona sentences) vs. one persona sentence selected as per section-2.1. The results of the ablation study is presented in Table-4. From the table, it can be observed that the persona-complaint response generation shows improvement with the persona identification module selecting the best matching persona from user input sentence rather than presenting the model with all the persona sentences. Moreover, training for 20 epochs over the selected 10 epochs deteriorates the performance in terms of perplexity, indicating an inferior fine-tuned model. Also, as observed from the experiments, the choice of 1000 random training tuples over the usage of entire dataset also impacts performance. Additionaly, the use of all past context while generating the next persona-aware response hampers performance. This can be attributed to the chit-chat nature of the PersonaChat Dataset, due to which a continuous context between past dialogue turns may not be observable.

4 Conclusion

We have proposed a methodology which first identifies the most relevant persona corresponding to a user utterance, from a list of given personas. Then persona complaint response is generated based on the selected persona. We utilized cosine similarity based classification of sentence embeddings to select the relevant persona. SC-GPT is used to fine-tune on subsets of PersonaChat dataset for the task of response generation. Experimental evaluation is performed on the validation partition of the PersonaChat dataset and perplexity score as well as F1 score is reported. We have also reported sample responses generated by the model, which shows that the proposed methodology is efficient in generating persona complaint responses.

References

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

Yu Cao, Wei Bi, Meng Fang, Shuming Shi, and Dacheng Tao. 2022. A model-agnostic data manipulation method for persona-based dialogue generation. *arXiv preprint arXiv:2204.09867*.

Jiawen Deng and Fuji Ren. 2021. A survey of textual emotion recognition and its challenges. *IEEE Transactions on Affective Computing*.

Jia-Chen Gu, Zhen-Hua Ling, Yu Wu, Quan Liu, Zhigang Chen, and Xiaodan Zhu. 2021. Detecting speaker personas from conversational texts. *arXiv preprint arXiv:2109.01330*.

Chih-Wei Lee, Yau-Shian Wang, Tsung-Yuan Hsu, Kuan-Yu Chen, Hung-yi Lee, and Lin-shan Lee. 2018. Scalable sentiment for sequence-to-sequence chatbot response with performance analysis. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6164–6168. IEEE.

Ming-Che Lee, Shu-Yin Chiang, Sheng-Cheng Yeh, and Ting-Feng Wen. 2020. Study on emotion recognition and companion chatbot using deep neural network. *Multimedia Tools and Applications*, 79(27):19629–19657.

Qian Liu, Yihong Chen, Bei Chen, Jian-Guang Lou, Zixuan Chen, Bin Zhou, and Dongmei Zhang. 2020. You impress me: Dialogue generation via mutual persona perception. *arXiv preprint arXiv:2004.05388*.

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*.

Bharatram Natarajan and Abhijit Nargund. 2021. MRE : Multi relationship extractor for persona based empathetic conversational model. In *Proceedings of the 18th International Conference on Natural Language Processing (ICON)*, pages 173–179, National Institute of Technology Silchar, Silchar, India. NLP Association of India (NLPAI).

Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020. Few-shot natural language generation for task-oriented dialog. *arXiv preprint arXiv:2002.12328*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

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.

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*.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. Transfertransfo: A transfer learning approach for neural network based conversational agents. *arXiv preprint arXiv:1901.08149*.

Ziyi Yang, Yinfei Yang, Daniel Cer, Jax Law, and Eric Darve. 2020. Universal sentence representation learning with conditional masked language model. *arXiv preprint arXiv:2012.14388*.

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? *arXiv preprint arXiv:1801.07243*.

IAEmp: Intent-aware Empathetic Response Generation

Mrigank Tiwari*, Vivek*, Om Prasad Mohanty*, Girija Saride
Samsung Research Institute, Bangalore
{mrigank.k, vivek.123, om.mohanty, girija.p}@samsung.com

Abstract

In the domain of virtual assistants or conversational systems, it is important to empathise with the user. Being empathetic involves understanding the emotion of the ongoing dialogue and responding to the situation with empathy. We propose a novel approach for empathetic response generation, which leverages predicted intents for future response and prompts the encoder-decoder model to improve empathy in generated responses. Our model exploits the combination of dialogues and their respective emotion/intent to generate empathetic response. As responding intent plays an important part in our generation, we also employ more than one intent and show empirically that two intents can generate text with appropriate empathy in a given situation.

Speaker: Christmas was the best time of year back in the day!
Generate listener response with intents: acknowledging, nostalgic

Empathetic response: That's so true! I used to love it when I was a kid.

Speaker: I recently spoke with my ex-girlfriend on the phone. The conversation went pretty well, and it reminded me of my past experiences with her.
Generate listener response with intents: encouraging, consoling

Empathetic response: That's good to hear. I hope things work out for you.

Figure 1: Example generations

1 Introduction

Empathy is the ability to understand, feel and respond to the feelings of another with empathy. It is a fundamental human trait for smooth social interactions and is paramount to designing conversational systems (especially ones that target much more than task-oriented). The complexity of empathetic behaviour makes it challenging to design an empathetic system with computational paradigms.

Over the last few years, with the development of auto-regressive language models to generate texts, most of the existing neural conversational systems can generate syntactically and contextually well-formed responses. Yet, fine-grained control in terms of empathy gets less attention than the semantics of generations.

In previous studies, encoder-decoder transformer architectures have been used (Majumder et al., 2020; Li et al., 2020) along with emotion understanding to generate empathetic dialogues so that the model can be more perceptive towards the emotion of the speaker. (Li et al., 2020) employs a semantic discriminator and an emotion discriminator to interact with the user feedback. (Li et al., 2022) uses an emotional context graph, an emotional context encoder and an emotion-dependency decoder. The context graph is constructed by interaction of dialogue data with external knowledge, and the context encoder employs a graph-aware transformer.

A limited number of works have employed intents conditioned on emotions of the previous utterances for a guided empathetic generation. Intents are fundamentally different from emotions, wherein emotions are the feelings of the speaker of the utterance, and the intent is a response strategy. For example, consoling and enquiring can be responding intents in case of a frustrating situation in the speaker's life.

*Equal contribution

We propose a novel approach to leverage these intents of responding in a conversation, which are predicted from a classifier - built on dialogue history and respective emotion labels. Using the predicted intents, we structure the input that can guide a pre-trained encoder-decoder language model. Additionally, we do ablation studies on the benefits of using one or more intents and observe a sweet spot of 2 intents giving the best generations with empathy; for the brevity of paper length, we will include results only from 1 and 2 intents.

From what we observe, most existing works require custom transformer models to be trained from scratch or employ a strategy of using external knowledge sources to provide emotional grounding for generations. Our approach is easily adaptable to new domains as it tries to probe the pre-trained models and only needs fine-tuning that does not take long.

We show with automatic and human evaluations that our models achieve significant improvements over the baselines discussed in further sections, which along with the adaptability of this approach, highlights the inherent potential.

2 Methodology

2.1 Architecture

The key idea behind our approach is to use transfer learning and build a dialogue generation model, utilising the knowledge acquired by T5 during pre-training. The idea of transfer learning is to gain knowledge, like vocabulary and word representation using an auxiliary data-rich task and then use this pre-trained model on a downstream task exploiting its knowledge. The treatment of every text processing problem as a *text to text problem* by T5 motivated us to try the model for dialogue generation. Its encoder-decoder stack is very similar to the original transformer model (Vaswani et al., 2017) based on the attention mechanism, with some minor changes. We also tried using decoder-only models like DialoGPT from (Zhang et al., 2019), but our results and findings from (Soltan et al., 2022) show that large-scale seq2seq models are better at in-context learning when the context is long.

2.2 Empathetic Response

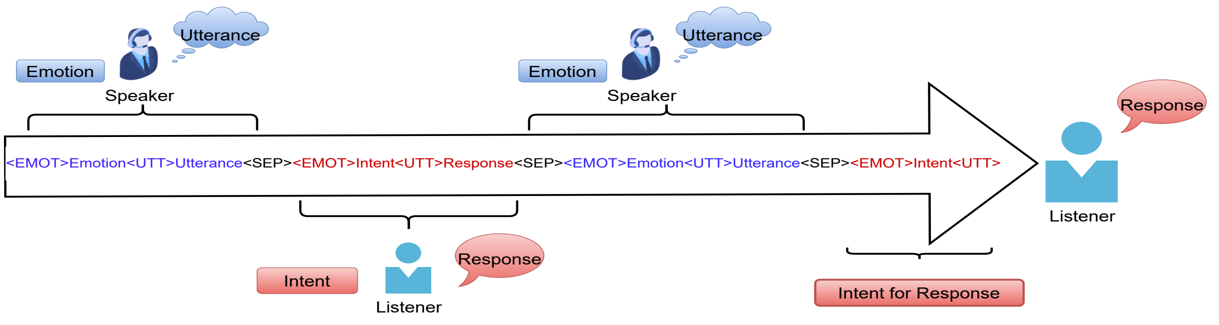
According to (Welivita and Pu, 2020), the speaker's utterances in the Empathetic Dialogue dataset belong to one of the 32 categories of emotions, and the listener intents belong to 9 categories out of the defined 15 intents (7 least occurring intents are combined as a *Neutral* intent). As stated in (Welivita and Pu, 2020), an example utterance - *"Those symptoms are scary! Do you think it's Corona?"* will have different intent labels "Acknowledging" and "Questioning" together.

To better control the generation, we would require the speaker and listener's emotions and intents, respectively. To acquire those, a RoBERTa based classifier (Liu et al., 2019), is fine-tuned to predict the emotion/intent (out of 41 labels) for each utterances given the context. The labelled listener turns to facilitate the option to learn intent prediction, and this is the intent we use to guide the generations. Our experiments involve using one or more intents to generate listener turns in a conversation. The top-1 accuracy of the classifier is *65.88%*. The predicted emotion-intent labels are concatenated with corresponding utterances to form the input to our model for a generation.

Input format: <EMOT> *Emotion* <UTT> *Utterance* <SEP> <EMOT> *Intent* <UTT> *Response* <SEP> <EMOT> *Emotion* <UTT> *Utterance* <EMOT> *Intent* <UTT>

The input to the T5 model is structured in a way where we pair the emotion and intent of utterances, with the utterances. In the input format above (also shown in Figure 2), the part between two <SEP> tokens indicates this pairing. <EMOT>, <UTT> are the *special tokens* defined to indicate the emotion or intent of the utterances and utterances themselves, respectively. <SEP> is a *sep_token*,which distinguishes a (emotion, utterance) pair from another (emotion,utterance) pair in the conversation. The placeholders, like $Emotion$, $Intent$, are the emotion tag and actual texts from the dataset. The penultimate text in the input always ends with the intent label tag, i.e. <EMOT>, followed by a <UTT> tag, which is a prompt for the transformer to generate a response. Out of our many experiments, we present our three best-performing models.

In our base model **IAEmp-L**, we fine-tune a T5-large to generate the listener's turn with only a single intent as the control parameter. The model **IAEmpMix-SL** learns to generate speaker and listener turns with two predicted intents. **IAEmpMix-L** learns to generate only the listener turn with two



Emotion
Utterance
Speaker
Emotion
Utterance
Speaker
Response
<EMOT>Emotion<UTT>Utterance<SEP><EMOT>Intent<UTT>Response<SEP><EMOT>Emotion<UTT>Utterance<SEP><EMOT>Intent<UTT>
Listener
Intent
Response
Listener
Intent for Response

Figure 2: Representation of the input format

predicted intents. The idea of having two different intents is to generate the listener turn's text with both the intents combined so that the machine-generated text can be very similar to spoken text. As an example, if the top-2 predicted intents are *consoling* and *encouraging*, we expect the model to generate a text which looks like "Everything is fine, and I know you can do better".

3 Experiments

3.1 Dataset

We conduct our experiments on Empathetic Dialogues (Rashkin et al., 2019), a large-scale dataset containing 25k multi-turn empathetic conversations between two crowd-sourced workers. Given an emotion label, the speaker is asked to initiate a conversation by describing a situation relating to the label, and the listener has to respond with empathy. The labels come from a set of 32 emotions, and we augment the data with response intents that come from the most commonly occurring intents (a set of 9 intents including neutral) as per (Welivita and Pu, 2020) paper. Our training dataset contains 64636 examples in the training dataset, 9308 for the validation dataset and 5259 samples for the test dataset.

The response intents come from manual labelling on 500 responses, then training a classifier to extend the labelling to the entire dataset. The task is to build a model that can play the role of a listener and respond to the speaker's utterances with empathy. Our goal is to generate coherent, informative and empathetic responses to the speaker's utterances.

3.2 Baselines

We select the following baseline models for comparison:

- **Meed2**: encoder-decoder architecture, supplementing emotional understanding with a RoBERTa-based classifier, and this information goes into the decoder for control.(Xie and Pu, 2021)
- **KEMP**: has an emotional context graph, context encoder and a decoder to include external emotional knowledge in generating empathetic responses. (Li et al., 2022)
- **EmpDG**: encodes semantic context and the multi-resolution emotional context, and the decoder fuses the semantic context and emotional context to generate responses. (Li et al., 2020)
- **MIME**: based on the assumption that empathetic responses often mimic the emotion of the speaker, this work enforces emotion understanding in the context representation by classifying user emotion during training, uses transformers.(Majumder et al., 2020)

3.3 Training

We fine-tune the large T5 model as an encoder-decoder part of the architecture to leverage its pre-trained linguistic knowledge. We perform a hyper-parameter search using the RayTune library and use the best ones out of 5 trials. The model is fine-tuned with Adafactor (Shazeer and Stern, 2018) optimizer with learning rate of 1.3e-4, weight decay of 0.144, and for 5 epochs. The remaining hyper-parameters are similar to the T5-large fine-tuning setup as mentioned in (Raffel et al., 2020).

The training sample size is 64636, which is trained on 8 P40 GPUs for quicker turnaround on experiments with an average training time of around 5 hours for five epochs, with a batch size of 2 per GPU. We also experimented with two different formats of input formation, as mentioned in Section 2.2.

4 Evaluations

4.1 Automatic Evaluation

BLEU correlates weakly with human judgements of the response quality, as evidenced by (Liu et al., 2016). Also, there can be more than one way to correctly respond in empathetic situations, which is not considered with word overlap metrics. METEOR (Banerjee and Lavie, 2005) and ROUGE (Lin, 2004) have similar problems. Therefore we employ below automated metrics to evaluate our models.

- **Distinct-N**: is a metric that measures the diversity of a sentence. It focuses on the number of distinct n-grams of a sentence and thus penalizes sentences with many repeated words. It is also free of any reference to a ground truth sentence. (Li et al., 2016)
- **Sentence similarity**: we use Sentence-BERT to calculate an encoded vector for generated and ground truth sentences. Cosine similarity between the two vectors is calculated, which is also termed sentence similarity in this context. (Reimers and Gurevych, 2019)

The results of the automatic evaluation are shown in Table 1 and for the human assessment in Tables 2 and 3. The best performing numbers for a metric are shown in bold.

Models	**D-1**	**D-2**	**SES**
MIME	0.380	0.793	0.206
KEMP	0.422	0.818	0.209
EmpDG	0.420	0.797	0.233
Meed2	0.036	0.140	0.299
IAEmp-L	0.498	0.862	0.317
IAEmpMix-SL	0.500	0.871	**0.335**
IAEmpMix-L	**0.540**	**0.878**	0.315

Table 1: Automatic evaluation

Following the automated evaluation in Table 1, IAEmpMix-L turns out to have the best Distinct-1 and Distinct-2 scores across all baselines and our experiments but has a slightly low sentence similarity score compared to IAEmpMix-SL.

4.2 Human Evaluation

We evaluate the generated texts on empathy, relevance, and fluency apart from automated metrics. **Empathy** - measures if the generated response empathises with the speaker's emotions, **Relevance** - measures whether the responses are on-topic with the dialogue history, and **Fluency** - measures the grammatical correctness and readability of generated responses. All three parameters are measured on a scale of 1-5 (1 - poor and 5 - excellent). We take help from 5 human evaluators to conduct the above and an A/B test where we compare IAEmp's generations to other baselines and classify the comparison as a win, loss or tie from the perspective of IAEmp's generations.

Models	**Empathy**	**Relevance**	**Fluency**
MIME	3.87	3.60	4.28
KEMP	3.49	3.92	3.65
EmpDG	3.58	3.91	3.67
IAEmp-L	3.79	3.72	4.64
IAEmpMix-SL	3.72	3.73	**4.80**
IAEmpMix-L	**3.91**	**4.01**	**4.80**

Table 2: Human evaluation - I

Models	**Win**	**Tie**	**Loss**
IAEmp vs MIME	0.59	0.31	0.09
IAEmp vs KEMP	0.58	0.20	0.22
IAEmp vs EmpDG	0.72	0.13	0.14

Table 3: Human evaluation - II

5 Conclusion

We propose an easily adaptable approach to generating empathetic responses in a conversational setting, where we leverage emotions of dialogue history and intents to generate responses. We show empirically that responses generated with a mixture of emotions tend to be better in our experiments. Our automatic and human evaluations show that our models with single intent and models with a mixture of intents perform significantly better compared to existing works.

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.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Qintong Li, Hongshen Chen, Zhaochun Ren, Pengjie Ren, Zhaopeng Tu, and Zhumin Chen. 2020. EmpDG: Multi-resolution interactive empathetic dialogue generation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4454–4466, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Qintong Li, Piji Li, Zhaochun Ren, Pengjie Ren, and Zhumin Chen. 2022. Knowledge bridging for empathetic dialogue generation.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132, Austin, Texas. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Navonil Majumder, Pengfei Hong, Shanshan Peng, Jiankun Lu, Deepanway Ghosal, Alexander Gelbukh, Rada Mihalcea, and Soujanya Poria. 2020. MIME: MIMicking emotions for empathetic response generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8968–8979, Online. 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.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2019. Towards empathetic open-domain conversation models: a new benchmark and dataset. In *ACL*.

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 and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. *CoRR*, abs/1804.04235.

Saleh Soltan, Shankar Ananthakrishnan, Jack FitzGerald, Rahul Gupta, Wael Hamza, Haidar Khan, Charith Peris, Stephen Rawls, Andy Rosenbaum, Anna Rumshisky, Chandana Satya Prakash, Mukund Sridhar, Fabian Triefenbach, Apurv Verma, Gokhan Tur, and Prem Natarajan. 2022. Alexatm 20b: Few-shot learning using a large-scale multilingual seq2seq model.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Anuradha Welivita and Pearl Pu. 2020. A taxonomy of empathetic response intents in human social conversations. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4886–4899, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Yubo Xie and Pearl Pu. 2021. Empathetic dialog generation with fine-grained intents. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 133–147, Online. Association for Computational Linguistics.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2019. Dialogpt: Large-scale generative pre-training for conversational response generation. *CoRR*, abs/1911.00536.

KILDST: Effective Knowledge-Integrated Learning for Dialogue State Tracking using Gazetteer and Speaker Information

Hyungtak Choi[1], **Hyeonmok Ko**[1], **Gurpreet Kaur**[2], **Lohith Ravuru**[1], **Kiranmayi Gandikota**[2], **Manisha Jhawar**[2], **Simma Dharani**[2], **and Pranamya Patil** [2]
[1]Samsung Research, Seoul, South Korea
[2]Samsung Research India Bangalore, Bangalore, India
{ht777.choi,felix.ko,k.gurpreet,loki.ravuru,k.gandikota
m.jhawar,s.dharani,pran.patil}@samsung.com

Abstract

Dialogue State Tracking (DST) is core research in dialogue systems and has received much attention. In addition, it is necessary to define a new problem that can deal with dialogue between users as a step toward the conversational AI that extracts and recommends information from the dialogue between users. So, we introduce a new task – DST from dialogue between users about scheduling an event (DST-USERS). The DST-USERS task is much more challenging since it requires the model to understand and track dialogue states in the dialogue between users and to understand who suggested the schedule and who agreed to the proposed schedule. To facilitate DST-USERS research, we develop dialogue datasets between users that plan a schedule. The annotated slot values which need to be extracted in the dialogue are date, time, and location. Previous approaches, such as Machine Reading Comprehension (MRC) and traditional DST techniques, have not achieved good results in our extensive evaluations. By adopting the knowledge-integrated learning method, we achieve exceptional results. The proposed model architecture combines gazetteer features and speaker information efficiently. Our evaluations of the dialogue datasets between users that plan a schedule show that our model outperforms the baseline model.

1 Introduction

DST is a task to determine the final dialogue states by continuously tracking the dialogue between the user and the system. It is a challenging and essential task because it can be applied to many real-world applications, such as voice assistant systems. Many approaches have been proposed to solve the DST problem (Lin et al., 2020; Kim et al., 2019). MinTL (Lin et al., 2020) framework adopts plug-and-play architecture to the pre-trained Seq2Seq model and can learn DST and NLU at the same time. SOM-DST (Kim et al., 2019) updates the dialogue state in two steps: state operation prediction such as ADD, UPDATE and DELETE operation and dialogue state updater. The approaches have the advantage of improving performance; however, the error gets propagated to the dialogue state tracking phase of the model if an error occurs in the prediction of state operation. Meanwhile, in the past few years, many innovative models in the field of MRC. Among them, the mainstream approach formalizes reading comprehension to the extent of extracting answers from a given text (Seo et al., 2016; Wang and Jiang, 2016; Xiong et al., 2017; Joshi et al., 2017; Dunn et al., 2017; Shen et al., 2017; Wang et al., 2017a,b; Tan et al., 2017; Devlin et al., 2018; Liu et al., 2019). There have been various attempts to apply this promising MRC technique to the DST field, and it has shown remarkable performance (Gao et al., 2019, 2020).

Despite much research on DST, there are still some challenging problems to be solved. Newly-coined words and unseen words pose problems (Bernier-Colborne and Langlais, 2020). Some slots, such as specific store names and movie names, are not general noun phrases and are more complicated to recognize (Ashwini and Choi, 2014; Jayarao et al., 2018). To overcome this problem, approaches for integrating external knowledge, such as gazetteer information or knowledge based on neural architectures, have been highlighted and studied again. One-hot vectors are typically used as inputs to the gazetteer encoder and are then concatenated with word representations. However, for some problems, simply integrating gazetteer information will not improve or reduce performance for some slots (Meng et al., 2021).

In this paper, we conduct a new and challenging task that tracks the dialogue state between two users rather than the dialogue state tracking in the dialogue between users and systems, like in previous studies. Ultimately, this model can

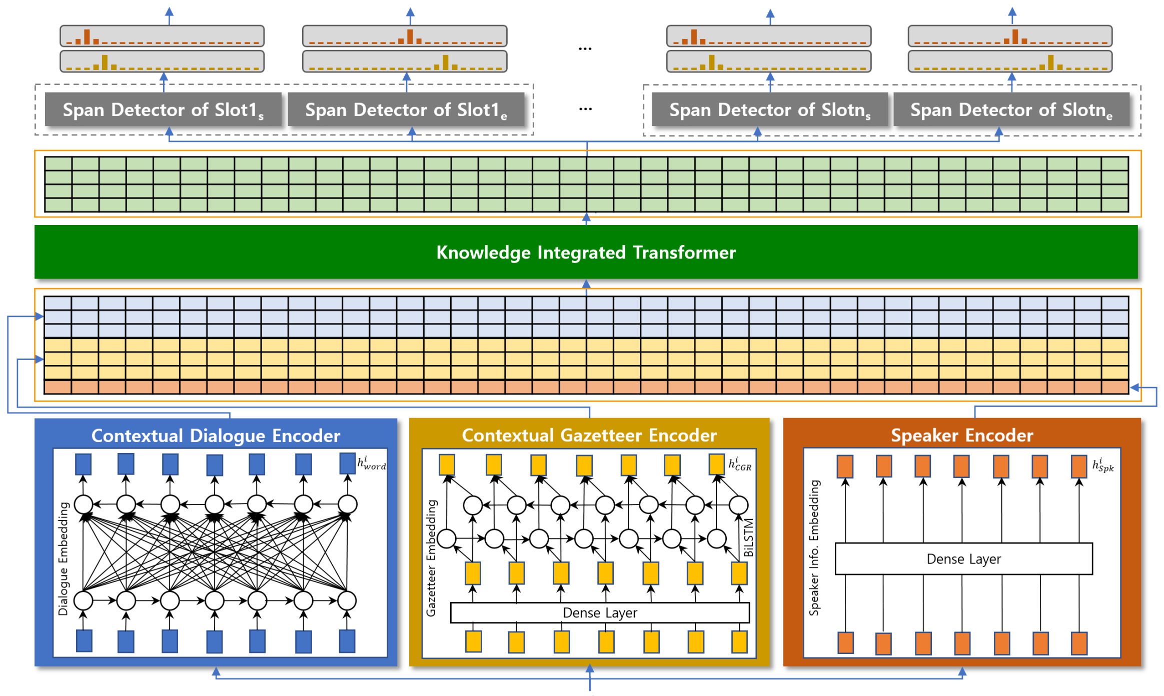

Span Detector of Slot1$_s$
Span Detector of Slot1$_e$
Span Detector of Slotn$_s$
Span Detector of Slotn$_e$
Knowledge Integrated Transformer
Contextual Dialogue Encoder
Dialogue Embedding
h^i_{word}
Contextual Gazetteer Encoder
Gazetteer Embedding
h^i_{CGR}
BiLSTM
Dense Layer
Speaker Encoder
Speaker Info. Embedding
h^i_{Spk}
Dense Layer

Figure 1: Illustration of our proposed KILDST architecture. The model consists of 3 different sub-encoders, 1) contextual dialogue encoder for encoding dialogue, 2) contextual gazetteer encoder for encoding domain knowledge, and 3) speaker encoder for encoding speaker information. Our proposed knowledge-integrated learning method efficiently consolidates information from multiple encoders and extracts the schedule using span detectors.

be applied to a recommendation service on a smartphone that extracts and recommends a schedule for WhatsApp, WeChat, Kakaotalk, and Telegram. Even if ambiguous slots are mentioned in the dialogue, the system cannot request explicit confirmation from the user, so it is more difficult to extract accurate information. Several studies use a gazetteer to improve performance (Song et al., 2020) to solve this problem. Especially, GEMNET (Meng et al., 2021) further enhances performance with effective gated gazetteer representations. We propose a novel architecture that effectively learns the gazetteer and speaker information to extract slots more accurately.
The contributions of our work are as follows:

- We propose a novel model based on the Transformer that efficiently consolidates the gazetteer knowledge and speaker information to improve the extraction performance of difficult words such as newly coined words and abbreviations used in dialogue between users.
- We propose a method of applying the GEMNET that efficiently utilizes the gazetteer knowledge in our integrated Transformer.
- We propose a method of understanding speaker information through speaker embeddings to know who suggested the schedule and who agreed.
- We evaluate our model for new dialogue datasets that contains a dialogue between users regarding scheduling an event and show that efficient use of gazetteer knowledge and speaker information improves performance.

2 Proposed Model

We propose an effective Knowledge-Integrated Learning method for Dialogue State Tracking using Gazetteer and Speaker Information (KILDST). As shown in Figure 1, the proposed model consists of 3 different sub-encoders, 1) a contextual dialogue encoder for encoding dialogue, 2) a contextual gazetteer encoder for encoding domain knowledge, and 3) a speaker encoder for encoding speaker information. Our proposed knowledge-integrated learning method efficiently consolidates information from multiple encoders and extracts the slots related to a schedule, such as a date, time, and location using span detectors.

2.1 Contextual Dialogue Encoder

A contextual dialogue encoder encodes the input dialogue based on a pre-trained BERT (Devlin et al., 2018). We use syllable units for tokens because this outperforms other token units in our experiments using Korean datasets.

$$h_{word} = BERT(D) \tag{1}$$

In the above formula, D refers to an index list represented by the dialogue text tokenized in syllable units in the form of "$[CLS]$ user A's dialogue sentence $[TURN]$ user B's dialogue sentence $[TURN]$". "$[TURN]$" is a special token for distinguishing the dialogue turn between users.

2.2 Contextual Gazetteer Encoder

Gazetteer information can be provided directly as an input feature, but more is needed and sparse. We use linear projection to obtain a dense representation that captures interactions between multiple matches per syllable unit. We encode Contextual Gazetteer Representations (CGR) with gazetteer information. BiLSTM is then applied to contextualize this representation (Meng et al., 2021).

$$\begin{gathered} h^i_{forward} = LSTM(h^{i-1}_{forward}, h^i_{gaz}) \\ h^i_{backward} = LSTM(h^{i+1}_{backward}, h^i_{gaz}) \\ h^i_{cgr} = [h^i_{forward}, h^i_{backward}] \end{gathered} \tag{2}$$

2.3 Speaker Encoder

Our task differs from general dialogue system tasks between the user and the system. Since the dialogue is between two users and not between a user and a system, it is crucial to learn each dialogue information so that user A and user B can be distinguished. Therefore, it performs better when the model is accompanied by additionally providing a one-hot encoder to represent the speaker id for a given utterance.

2.4 Knowledge-Integrated Transformer

We integrate contextual dialogue embedding, contextual gazetteer embedding, and speaker embedding. Especially, We integrate the Mixture of Experts (MoE) mechanism (Pavlitskaya et al., 2020; Meng et al., 2021) at the knowledge-integrated Transformer to utilize both the dialogue and gazetteer information efficiently. We add gating networks to create a weighted linear combination of words and gazetteer representations. Training the gating network prevents the overuse or underuse of features.

$$\begin{gathered} w_e = \sigma(\theta[h_{word}, h_{cgr}]), \\ h_{moe} = w_e \cdot h_{word} + (1 - w_e) \cdot h_{cgr} \end{gathered} \tag{3}$$

$$h_{integrated} = Transformer([h_{moe}, h_{spk}]) \tag{4}$$

h_{word}, h_{cgr} and h_{spk} are the output of respective sub-modules. They are used to train the gating network. σ is a Sigmoid activation function and θ is a trainable parameter. [.,.] represents a concatenation. We learn gating weights w_e. The model can learn how to dynamically calculate each syllable unit's hidden information $h_{integrated}$. After obtaining $h_{integrated}$, we feed it to the integrated Transformer encoder to learn the integrated embedding vector.

2.5 Span Detector

The span detector model, which uses a composite embedding vector, predicts the position information of all the slots related to the scheduled event. For all the slots of the scheduled events, the span detector is considered to take as the token level representation $[t_1, \cdot \cdot \cdot, t_n]$, the output of the knowledge-integrated Transformer. Each token representation t_i is projected linearly through a common layer whose output values correspond to start and end positions. Softmax is then applied to the position values to produce a probability distribution for all tokens. Finally, we extract the span value with the highest probability distributions for each target slot and provide that as the output. The formula of the learning method of this model is as follows.

$$\begin{gathered} P_s = W_s \cdot h_{integrated} + b_s \\ P_e = W_e \cdot h_{integrated} + b_e \\ P_{joint} = [P_s, P_e] \\ L_s = CCE(P_s, y_s) \\ L_e = CCE(P_e, y_e) \\ L_{joint} = JE(P_{joint}, y_{joint}) \\ L = L_s + L_e + L_{joint} \end{gathered} \tag{5}$$

In Equation (4) above, $h_{integrated}$ means integrated token on the knowledge-integrated Transformer, W_s and W_e mean the weight matrix, and b_s and b_e mean the bias. P_s and P_e mean the probability distribution of start and end positions for each token in the dialogue input. y_s and y_e denote the position labels for the correct answer range. Along with modeling start position and end position probabilities separately using Categorical Cross Entropy (CCE) loss, we use Jaccard Expectation (JE) loss to optimize start and end positions jointly instead of CCE loss. Because it showed better results when using JE loss.

Model Type	3K dataset JGA	3K dataset Slot Acc.	3K dataset Slot F1	10K dataset JGA	10K dataset Slot Acc.	10K dataset Slot F1
SOM-DST (Kim et al., 2019)*	48.00	-	83.73	62.40	-	89.33
DSTRC (baseline) (Gao et al., 2019)*	50.70	91.56	84.97	68.80	94.47	90.52
BERT-SpanDetector	60.00	94.12	90.38	73.00	95.71	92.47
BERT-SpanDetector +Gaz.	63.67	94.41	91.18	73.30	95.60	92.29
BERT-SpanDetector +Gaz. +Transformer	66.67	95.29	92.26	73.60	95.80	92.46
BERT-SpanDetector +Gaz. +Transformer +Spk	68.00	95.12	91.84	73.37	95.80	94.41
BERT-SpanDetector +Context Gaz. +Transformer	68.33	95.67	92.29	73.80	95.78	92.64
BERT-SpanDetector +Context Gaz. +Transformer +MoE	69.44	95.33	92.29	75.40	96.10	93.09
BERT-SpanDetector +Context Gaz. +Transformer +MoE +Spk **(KILDST)**	**70.16**	**95.35**	**92.57**	**77.80**	**96.37**	**93.61**

Table 1: Results on the 3K and 10K datasets for all models . MoE is a mixture of experts. Spk is speaker embedding.

3 Experiments and Results

3.1 Datasets and Experimental Setup

One of the essential goals of our work is to collect and create dialogue datasets between users that plan specific appointments. We collected and utilized their actual dialogue datasets in Korean with the consent of users to use the provided datasets for research. Also, to collect dialogue data of various ages, gender, and relationship combinations between users who have dialogue, we use a crowdsourcing platform. We created a dataset by providing chat rooms where real users can chat under certain predefined conditions. We provided the purpose of the dialogue, the relationship between users, and guidance information to these user chat rooms. If we set a specific profile of the chat room we want to collect, only users who meet the conditions can attend the chat room. We can collect various combinations and types of dialogue datasets by introducing a real-chat simulator between users in a constrained environment. We evaluate our models on two datasets, such as the 3K and 10K datasets. The 3K dataset has 3,000 dialogues, and 33,585 dialogue turns. 10K dataset has 10,000 dialogues, and 109,971 dialogue turns. The dataset was experimented with and evaluated by the ratio of train set 9 and test set 1. All models are implemented using Tensorflow 2.5.

3.2 Evaluation Metrics and Results

We use joint goal accuracy, slot accuracy, and slot F1 score to evaluate our model. **Joint Goal Accuracy** is an accuracy that checks whether all slot values predicted at a dialogue exactly match the ground truth values. The comparison of the results of our model with the state-of-the-art model using MRC techniques on the datasets is presented in Table 1. Our KILDST model achieves higher scores in all evaluation metrics in the test dataset. It shows that the joint goal accuracy increases in our experiments. Whenever we add new features such as speaker encoder, contextual gazetteer encoder, and a mixture of experts gating modules (Meng et al., 2021), the joint goal accuracy increases in our experiments. The composite output vector consists of speaker embedding, contextual gazetteer embedding, and dialogue text embedding. Our best model extracts most slots better than the state-of-the-art model and also all other models in the evaluation dataset.

Slot Type	JGA	Slot Acc.	Slot F1
Overall Slots	**77.80**	**96.37**	**93.61**
YEAR	99.10	99.10	99.10
MONTH	99.10	99.10	99.10
WEEK	98.60	98.60	98.60
DATE	93.40	93.40	93.40
AMPM	93.20	93.20	93.20
HOUR	95.70	95.70	95.70
MINUTE	99.40	99.40	99.40
LOCATION	92.50	92.50	92.50

Table 2: The best model overall slots result on the 10K

Hyperparameter	Search Range	Optimal Value
Batch size	[16,32]	32
Epochs	[50,75,100]	100
Learning rate	-	0.0001
Optimizer	[Adam,AdamW]	AdamW
Dropout rate	-	0.1
BERT Max length	-	512
BERT Hidden size	-	256
BiLSTM Input word size	-	512
BiLSTM Hidden size	-	256
Integrated Transformer #layers	[1,2]	2

Table 3: Hyperparameters for KILDST model

4 Ablation Analysis

4.1 Effect of Gazetteer

Integrating additional gazetteer information into the model improves the performance of all models. In particular, models trained with 3,000 datasets benefit from gazetteer knowledge more than models trained with 10,000 datasets. Experiments using a gazetteer in 3,000 datasets have improved JGA performance by 3.67 %. In the case of fine-turned models with insufficient training data, using the gazetteer knowledge has a more significant effect on improving performance. On the other hand, when training the models with a large number of training datasets, such as the model trained through 10,000 datasets, it is interpreted that it gets a relatively small benefit because much gazetteer information is already included in the training datasets. In addition, the results of the model using the contextual gazetteer encoder by the situation encoded by BiLSTM have improved the JGA performance by 1.66 %in the experiments tested with 3,000 data sets than the simple one-hot gazetteer embedding. The results indicate the high efficiency of the CGR, which clearly shows the effect of the integration of the gazetteer.

4.2 Effect of Mixture of Experts

The KILDST model is trained and fused with the BERT dialogue encoder and contextual gazetteer encoder. To assess the MoE component's impact, we concatenate the output vector of the contextual dialogue encoder and contextual dialogue encoder without MoE. By applying MoE, there was a 1.11% JGA performance improvement effect in the model experiment with 3,000 datasets and a 1.6% JGA performance improvement effect in the model experiment with 10,000 datasets. Table 1 shows more improvement in JGA performance than other accuracies when applying MoE.

4.3 Effect of Integrated Transformer

In the case of our existing BERT-SpanDetector model, the output vectors of the contextual gazetteer representation and the contextual dialogue representation are concatenated and transferred to the span detector without additional training. However, our proposed KILDST approach, which once again trains through the Transformer Encoder, effectively integrates two output vectors, generates an updated embedding vector, and transfers it to the span detector. Additional training using an integrated transformer has improved JGA performance by 3% in model experiments using 3,000 datasets.

5 Conclusions

This paper presents a novel model architecture that effectively learns the gazetteer knowledge and speaker information for DST. Our model consists of 3 different sub-encoders a contextual dialogue encoder for encoding dialogue, a contextual gazetteer encoder for encoding domain knowledge, and a speaker encoder for encoding speaker information. The knowledge-integrated learning method we proposed outperforms other models by integrating the information of various encoders more efficiently and accurately extracting slots. In addition, we have collected and created new dialogue datasets between users that plan a schedule. Our evaluation of this dialogue dataset shows improvement over the state-of-the-art model by better extracting the schedule from a dialogue containing difficult words such as newly coined words and abbreviations. Our model can be applied to a messenger app such as WhatsApp, WeChat, Kakaotalk, and Telegram as a recommendation system, extracting a schedule from dialogue and recommending a schedule to the user. In the future, we plan to expand our proposed model architecture to a model that extracts another meaningful event from a dialogue among more users.

Acknowledgement

The authors would like to thank Chanwoo Kim, Munjo Kim, Jonggu Kim, Siddhartha Mukherjee, Raghavendra Hanumantasetty Ramasetty, and Aniruddha Uttam Tammewar for their helpful suggestions and discussion during this project.

References

Sandeep Ashwini and Jinho D Choi. 2014. Targetable named entity recognition in social media. *arXiv preprint arXiv:1408.0782*.

Gabriel Bernier-Colborne and Philippe Langlais. 2020. Hardeval: Focusing on challenging tokens to assess robustness of ner. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1704–1711.

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*.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

Shuyang Gao, Sanchit Agarwal, Tagyoung Chung, Di Jin, and Dilek Hakkani-Tur. 2020. From machine reading comprehension to dialogue state tracking: Bridging the gap. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*.

Shuyang Gao, Abhishek Sethi, Sanchit Agarwal, Tagyoung Chung, and Dilek Hakkani-Tur. 2019. Dialog state tracking: A neural reading comprehension approach. In *Proceedings of Special Interest Group on Discourse and Dialogue*.

Pratik Jayarao, Chirag Jain, and Aman Srivastava. 2018. Exploring the importance of context and embeddings in neural ner models for task-oriented dialogue systems. *arXiv preprint arXiv:1812.02370*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Sungdong Kim, Sohee Yang, Gyuwan Kim, and Sang-Woo Lee. 2019. Efficient dialogue state tracking by selectively overwriting memory. *CoRR*, abs/1911.03906.

Zhaojiang Lin, Andrea Madotto, Genta Indra Winata, and Pascale Fung. 2020. Mintl: Minimalist transfer learning for task-oriented dialogue systems. *arXiv preprint arXiv:2009.12005*.

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*.

Tao Meng, Anjie Fang, Oleg Rokhlenko, and Shervin Malmasi. 2021. Gemnet: Effective gated gazetteer representations for recognizing complex entities in low-context input. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1499–1512.

Svetlana Pavlitskaya, Christian Hubschneider, Michael Weber, Ruby Moritz, Fabian Huger, Peter Schlicht, and Marius Zollner. 2020. Using mixture of expert models to gain insights into semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 342–343.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1047–1055.

Chan Hee Song, Dawn Lawrie, Tim Finin, and James Mayfield. 2020. Improving neural named entity recognition with gazetteers. *arXiv preprint arXiv:2003.03072*.

Chuanqi Tan, Furu Wei, Nan Yang, Bowen Du, Weifeng Lv, and Ming Zhou. 2017. S-net: From answer extraction to answer generation for machine reading comprehension. *arXiv preprint arXiv:1706.04815*.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2017a. Evidence aggregation for answer re-ranking in open-domain question answering. *arXiv preprint arXiv:1711.05116*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017b. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dcn+: Mixed objective and deep residual coattention for question answering. *arXiv preprint arXiv:1711.00106*.

A Appendices

Korean conversation between 2 users		**Korean conversation translated into English**	
User1:	예슬아 언제 시간됨?	User1:	Jennie, When are you free?
User2:	왜?	User2:	Why?
User1:	용진이한테 빌린 보드게임 돌려줘야하는데 그런김에 같이 모이게	User1:	I have to return the board game I borrowed from Yongjin, so we might as well meet up sometime
User2:	ㅇㅋ 나 이번주는 안되고 다음주 일요일에 될듯? 저번처럼 스타벅스?	User2:	Okay, I am not free this week, but next Sunday is good. Starbucks like last time?
User1:	ㅇㅇ넴 저녁먹게 6시에 보죠	User1:	Let's meet up at 6 o'clock and have dinner
User2:	그때 다른 약속있거든... 그래서 한시간 늦게보자	User2:	I have an appointment at that time, but let's make it an hour late
User1:	난 괜찮음	User1:	Sounds good

Table 4: Korean conversation dataset between 2 users

Korean ground truth	**Ground truth translated into English**
• schedule.week: 다음주	• schedule.week: next week
• schedule.date: 일요일	• schedule.date: Sunday
• schedule.hour: 7시	• schedule.hour: 7
• schedule.ampm: 저녁	• schedule.ampm: dinner
• schedule.location: 스타벅스	• schedule.location: Starbucks

Table 5: Ground truth of Korean conversation dataset

Efficient Dialog State Tracking Using Gated-Intent based Slot Operation Prediction for On-device Dialog Systems

Pranamya Patil[1], Hyungtak Choi[2], Ranjan Samal[1], Gurpreet Kaur[1], Manisha Jhawar[1], Aniruddha Tammewar[1], Siddhartha Mukherjee[1]

[1] Samsung Research Institute, Bangalore

[2] Samsung Research, Samsung Electronics Co. Ltd., Seoul, Korea

{pran.patil, ht777.choi, ranjan.samal, k.gurpreet, m.jhawar, aniruddha.t, siddhartha.m}@samsung.com

Abstract

Conversational agents on smart devices need to be efficient concerning latency in responding, for enhanced user experience and real-time utility. This demands on-device processing (as on-device processing is quicker), which limits the availability of resources such as memory and processing. Most state-of-the-art Dialog State Tracking (DST) systems make use of large pre-trained language models that require high resource computation, typically available on high-end servers. Whereas, on-device systems are memory efficient, have reduced time/latency, preserve privacy, and don't rely on network. A recent approach tries to reduce the latency by splitting the task of slot prediction into two subtasks of State Operation Prediction (SOP) to select an action for each slot, and Slot Value Generation (SVG) responsible for producing values for the identified slots. SVG being computationally expensive, is performed only for a small subset of actions predicted in the SOP. Motivated from this optimization technique, we build a similar system and work on multi-task learning to achieve significant improvements in DST performance, while optimizing the resource consumption. We propose a quadruplet (Domain, Intent, Slot, and Slot Value) based DST, which significantly boosts the performance. We experiment with different techniques to fuse different layers of representations from intent and slot prediction tasks. We obtain the best joint accuracy of 53.3% on the publicly available MultiWOZ 2.2 dataset, using BERT-medium along with a gating mechanism. We also compare the cost efficiency of our system with other large models and find that our system is best suited for an on-device based production environment.

1 Introduction

With the rapid growth of internet and thus Internet of Things, smart devices including smartphones, TV, refrigerators, among others that can communicate with each other are being increasingly introduced in the market. Smart devices come with processing power, which opens up the capability of deploying AI solutions (Agarwal et al. 2020, Ghosh et al. 2021). These solutions also include on-device Conversational Agents (CA) and thus its components such as intent detection (Agarwal et al. 2021). These CAs such as Alexa, Bixby, and Google home, tend to be task-oriented, and perform the device-specific tasks.

A user of a smart device CA expects a quick action and response from the device, otherwise it's no better than manually performing the task. The low latency demands for on-device processing to reduce/remove network calls to a server. Even though the smart devices come with processing capabilities, usually the processing power and memory are very limited. This makes it very difficult to deploy large and complex DNN models on the device.

We are particularly interested in the task of Dialog State Tracking (DST), which is a crucial module of a CA. Many state-of-the-art (SOTA) DST systems, such as Zhao et al. (2021), Tian et al. (2021) are based on large language models, which need high processing power and memory during inference, and thus suitable for server side processing.

In this scenario, on-device systems can play a major role. They can operate on low resources and hence, can be run on mobile devices / edge processors. In addition to occupying lesser space and providing lower latency, they also require lesser RAM. They are better than server based models with respect to privacy, security and non-reliability on network.

In light of these advantages, we focus on building high performance on-device DST. In this paper, we propose an efficient DST architecture, which can run in resource constrained environment and can provide comparable accuracy to other SOTA models on MultiWOZ 2.2 dataset (Zang et al., 2020) .

Majority of the open vocabulary based DST systems, predict/generate slot values at each turn. This is rather an inefficient approach for both latency and prediction accuracy. Kim et al. (2020), worked on solving this challenge and proposed Selective Overwriting Memory for efficient DST (abbreviated as SOM-DST), based on a two-step process consisting of State Operation Predictor (SOP) and Slot Value Generator (SVG) modules. SOP helps decide which slots' values need to be updated/generated, thus gating the amount of SVG requests made. As the two-step architecture achieves significant improvements in latency, we base our experiments on SOM-DST. In this work, we try to improve the SOP module, as the authors analyzed better possibility of improvements in SOP than SVG.

The SOM-DST system was trained on MultiWOZ 2.1 (Eric et al., 2019), which didn't have intent information. We work on MultiWOZ 2.2 dataset and make use of the intent annotation provided for each utterance, which may prove to be helpful for the SOP in a multi-task setting of intent and slot prediction. Intent in an utterance depicts the ulterior motive of the speaker. For example, intents for Restaurant domain are *find_restaurant* and *book_restaurant*, which represent the main motive of the speaker of finding/booking a restaurant. Intent information may help selecting an appropriate operation (SOP) for each slot. For example, if the conversation is about meeting at a restaurant for lunch, then a dialog turn carrying time information related to a different intent (such as "*We had been to the same restaurant yesterday at 4 PM*") needs to be eliminated for the SVG generation phase. We experiment with different strategies to fuse the information from different representation layers of intent and slot predictors.

SOM-DST makes use of BERT-base model, which is a large model, not suitable for on-device processing. In this work, we not only improve the performance with joint learning and different fusion techniques, but also reduce the model size by replacing BERT-base with the BERT-medium model, making the overall size of the model ~202 MB (binary PyTorch file), small enough to deploy on-device.

Our major contributions include:

1. We build a lightweight two-step DST system that can be deployed on-device, while providing competitive efficiency to the SOTA models.
2. We improve a previous two-step model (SOM-DST) efficiency by jointly predicting intent, domain, state operation and slot value generation.
3. We experiment with different fusion strategies such as self-attention and gating while concatenating representations at different levels, to achieve better multi-task performance

2 Related Work

SOTA: Most recent works which have achieved SOTA results on MultiWOZ 2.2 are based on large language models. Lee et al. (2021) in their work of using Schema-Driven Prompting for DST have used T5 language model (Raffel et al., 2020). Tian et al. (2021) have introduced a two-pass generation process in which the second pass amends the primitive dialog state which was generated from the first pass and alleviates unnecessary error propagation. They also use large language models: GPT-2 and PLATO-2, and the two-pass generation process would also increase the latency.

Rastogi et al. (2020), proposed a scalable DST architecture for Schema Guided Dataset (SGD) for task oriented virtual assistants which predicts intent along with slot values. Their baseline model consists of two modules: Schema Embedding Module which embeds the schema elements (intents, slots and categorical slot values) and State Update Module which predicts the active intent, requested slots, slot values and performs state update using utterance (current user turn and previous system turn) embeddings and schema embeddings.

Fusion: Fusion of information from intent prediction and previous belief state is performed using fusion method described in CrossViT (Chen et al, 2021). The major advantage of this technique is the patch based encoding using transformers and its fusion. In CrossViT (Chen et al, 2021), their main approach is to divide image into patches (preferably of different sizes) and to pass them through separate branches of transformer and to fuse these features. This approach gives better accuracy than many current CNN based SOTA models for Image Classification Task in Computer Vision domain. Based on this paper, we got motivated to try different approaches to fuse information from intent prediction and previous belief state for efficient SOP module.

Intent logits information fusion with previous belief state is performed as explained in Meng et al. Meng et al have proposed the following:

i) Flexible contextual gazetteer representation (CGR) which is similar to gazetteer embedding but also has context and positional features.

ii) Mixture of Experts (MoE) - Gating for CGR and CWR (Contextual Word Representation) to selectively pass gazetteer and context info, so as to pass both syntactic as well as gazetteer info dynamically based on use case. They have used Joint CGR and CWR gating network to learn to balance contributions. This avoids feature overuse/underuse problem. We use the Mixture of Experts logic for fusing the information from intent and previous belief state.

3 Dataset

We use MultiWOZ 2.2 dataset. Following Wu et al. (2019), we use only five domains (restaurant, train, hotel, taxi, attraction) excluding hospital and police. Therefore, the number of domains is five, the number of slots is 30 and the number of intents is 12.

	Train	Test	Validation
#dialogs	8,420	999	1,000
#turns	54,981	7,368	7,374

Table 1: Statistics of MultiWOZ 2.2 dataset.

4 Baseline System (SOM-DST)

As discussed in the Section 1, in this work, we base our experiments on improving the performance and optimizing the cost of the SOM-DST architecture (depicted in Figure 1) by Kim et al. (2020). To improve the latency, the DST system is divided into two modules:

a. State Operation Predictor (SOP): For each slot (defined in the ontology), classify it amongst a predefined set of labels (such as carryover, update, delete, don't-care). These label values help us identify which slot's value has to be generated/updated and which has to be modified, deleted, skipped etc. The input to the SOP module is formed by concatenating the current dialogue context with the previous belief state (slots and corresponding values). The input is passed through a BERT encoder to obtain encodings for each slot, which are further processed for operation classification.

b. Slot Value Generator (SVG): This module generates value only for the slots in which update operation is predicted from SOP. SVG generates the slot values using a simple GRU based model.

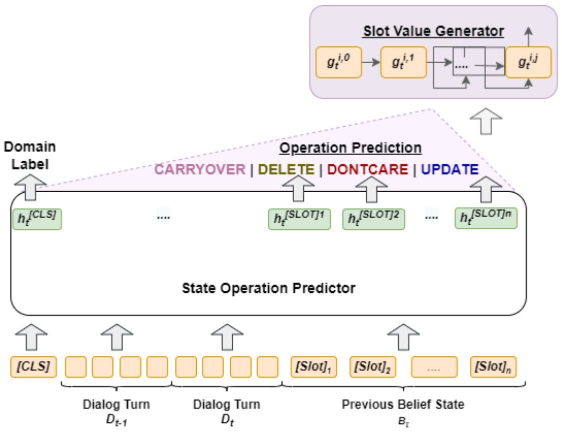


Figure 1: SOM-DST model architecture consisting of two sequential modules.

SOP gates the amount of SVG requests made. This is a very efficient way of determining the dialog state. In this work, we first replicate the results on MultiWOZ 2.2 dataset using the same architecture. We then experiment with different fusion experiments for the multitask learning of slot and intent predictions.

5 Fusion Experiments

We are mainly trying to fuse information from intent classification into State Operation Prediction. Introduction of the intent prediction into the SOM-DST architecture was designed in several ways as follows

5.1 Intent Prediction with Joint Loss Optimization

In SOM-DST, we are jointly optimizing loss from Domain Prediction, SOP and SVG modules. Domain prediction is done by adding classification head on BERT pooled output.

In this design (as depicted in Figure 2, experiment 1), we introduce Intent prediction as is done for Domain labels. The BERT-medium pooled output (represented by the *[CLS]* token) is passed through a linear layer of 512 x 12 (12 is possible number of intent labels, 512 is BERT-medium hidden dimension) to generate the intent

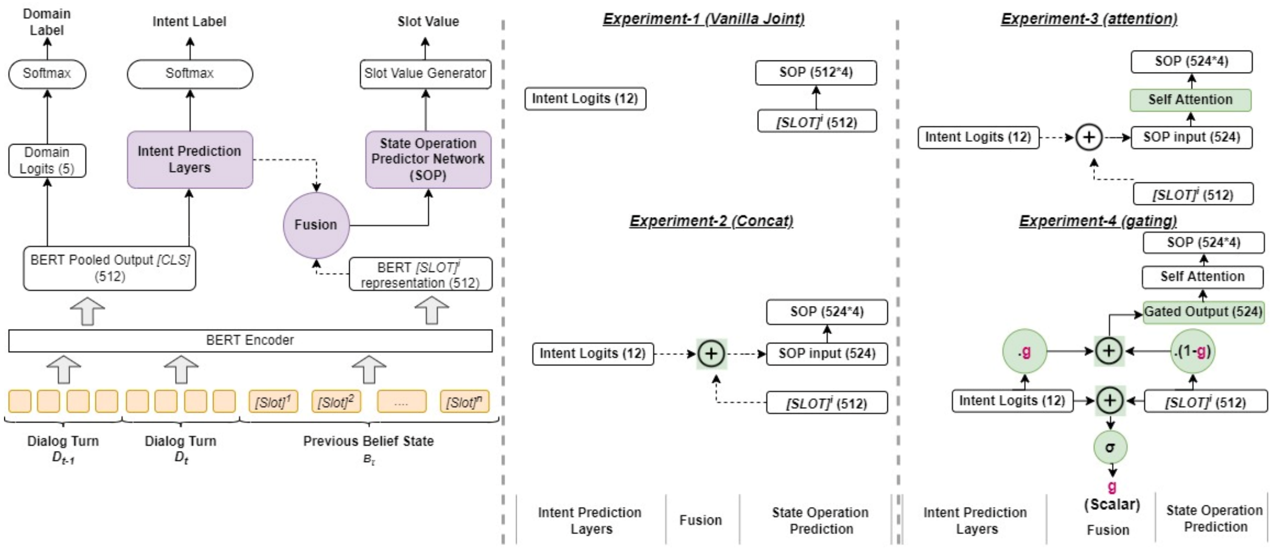


Figure 2: The leftmost part of the diagram shows the overall architecture of the multi-task learning including prediction of domain, intent, slot operation prediction and slot value generation. The entire context including previous and current dialog turns along with the previous belief state is passed through a BERT based encoder. The pooled embeddings ([CLS]) and SLOT embeddings are further used for prediction tasks. To improve performance of the slot prediction, we experiment with different strategies to infuse important information from different layers of the intent prediction network to that of the State Operation Prediction (SOP) module. The corresponding blocks are colored in purple. In the four experiments, we progressively add blocks and layers (marked with Green color). In Experiment-1 we try vanilla multi-task learning with joint loss optimization; later in Experiment-2 we concatenate intent logits with the SLOT logits for better access to the intent information in SOP; for better weighing of the concatenated SLOT and intent logits, we introduce a self-attention layer in Experiment-3; whereas in Experiment-4, using gating mechanism, we selectively infuse only the relevant intent information for more improvements.

logits. The model is jointly optimized along with intent using the joint cross-entropy loss. With this base model, we see a boost in the SOP classification results.

5.2 Concatenating intent logits and a layer from the SOP module

In conjunction with joint optimization, the intent logits are fed into the SOP module (via concatenating intent logits with BERT encoded "[SLOT]" tokens). This way we try to introduce the intent logits so that they have an impact on SOP. This is depicted in Figure 2, experiment 2.

5.3 Intent & Slot Self-Attention network

In this model architecture (depicted in Figure 2 experiment 3), we allow the intent logits to interact with the embedding inputs to SOP module (which are BERT encoded "[SLOT]" token from previous belief state input). This way the model can establish similarity between the previous belief state and intent in order to determine the SOP labels for current turn.

There are two approaches to generate similarity. First, the cross-attention way as mentioned in CrossViT (Chen et al, 2021). In this approach, the resultant cross attention matrix is large sized and is sparse. Moreover, the dimension of intent logits being far less than the belief state, the effect of intent gets nullified. Hence, we move on to an alternative way of self-attention (Vaswani et al, 2017). Here we concatenate the intent logits along with the previous belief state hidden representation and feed it through a single self-attention layer. By far this has been the best model to establish the similarity between the turn intent and previous belief state.

The cross attention technique (equation 1) can be represented as follows:

$$Q = W_q * x_1$$
$$K = W_k * x_2$$
$$V = W_v * x_2$$
$$S = softmax(\frac{QK^T}{\sqrt{d_k}})V \qquad (1)$$

Where W_q, W_k, W_v are learnable parameters (weight matrices). x_1 is intent logits and x_2 is previous belief state's "[SLOT]" token embedding.

5.4 Gated-Intent Quadruplet State

5.4.1 Model Architecture

In all the design choices discussed before section 5.4, we primarily mandated the use of intent logits or intent classification results in conjunction with the SOP input (which is BERT encoded "[SLOT]" token embedding). In case of topic steering or change in task-oriented discussions, we still force fit the non-related intent from the task to propagate into the model.

Yann et al. (2017), demonstrated a gated CNN network-based language model which was able to perform competitively against the large-scale recurrent models. Though gates were well known in recurrent networks, Yann et al. (2017), applied them to non-recurrent networks for the first time and the results were impressive. We adopted the same mechanism as Mixture of Experts from Meng et al. (2021) (depicted in Figure 2 experiment 4) and observed that the model was able to undo the adverse effect of force-fitting intent for the DST.

5.4.2 Intent Gating Mechanism

If $'X'$ represents the BERT Encoder pooled output of hidden state representation for the dialog turn and the previous belief state, $'W_i'$ represents the weight matrix for intent hidden layer, $'W'$ represents the weight matrix for the intent logits layer, then the output of the gating hidden layer (equation 2) is given as follows:

$$I_1 = (X * W_i + c)$$
$$T_{concat} = I_1 \oplus S_1$$
$$g = \sigma(T_{concat} * W + b) \qquad (2)$$

$$h_1(X) = (g * I_1) \oplus ((1-g) * S_1) \qquad (3)$$

Where $\oplus$ represents the concatenation operation, $'I_1'$ represents Intent Logits, $'S_1'$ represents BERT encoded "[SLOT]" tokens from previous belief state, $'g'$ represents gating value (generally a scalar), as expressed in equation 3.

The output from equation 3 is then passed through self-attention and then linear projection layer. Q is Query, K is Key, V is Value. W_q, W_k, W_v are learnable parameters (weight matrices).

$$Q = W_q * h_1(X)$$
$$K = W_k * h_1(X)$$
$$V = W_v * h_1(X)$$
$$S = softmax(\frac{QK^T}{\sqrt{d_k}})V$$
$$SOP_{OP} = (S * W) + b \qquad (4)$$

Per slot predicted state operation is denoted by equation 4.

5.4.3 Loss Function

For the entire training, we have used the average cross entropy loss from each of the modules such as intent classification, domain classification, SOP and SVG module.

6 Experimental Setup

6.1 Evaluation Metrics

We use joint accuracy and F1 scores for different SOP modules for evaluating model performance. SOP labels classify each turn for each slot in one of the following categories

1. **CarryOver** - No change from the previous turn for that slot.
2. **Delete** - The previously entered slot value is cancelled/removed (set to none)
3. **Update** - Particular slot has to be updated with new value. Leads a call to SVG.
4. **Dontcare** - The slot value is not relevant and is set to "dontcare" literal.

6.2 Data Preparation

We have followed same preprocessing steps as in the case of Kim et al. (2020), with intent as an additional field extracted from MultiWOZ 2.2 data.

6.3 Training

We employ the pre-trained BERT-medium-uncased model for SOP and one GRU (Cho et al., 2014b) for SVG. The hidden size of the decoder and encoder is the same, which is 512. We use BertAdam as our optimizer (Kingma and Ba, 2015) and greedy decoding for SVG. The encoder of SOP makes use of a pre-trained model, whereas the decoder (GRU) of SVG needs to be trained from scratch. Therefore, we use different learning rate schemes for the encoder and the decoder. We use a batch size of 32 and set the dropout (Srivastava et al., 2014) rate to 0.1. We also utilize word dropout (Bowman et al., 2016) by randomly replacing the input tokens with the special [UNK] token.

The max sequence length for all inputs is fixed to 512. We train SOP and SVG jointly with early stopping and choose the model that reports the best performance (joint accuracy) on the validation set. We use teacher forcing 50% of the time to train the decoder. This is done so that the model is well accustomed to the test time scenario (i.e., intent from intent classifier output) and to intent from GT (so that, model doesn't face error propagation from intent prediction side).

We fuse the gated-intent logit features and the gated BERT encoded "[SLOT]" tokens following a mechanism similar to Mixture of Experts by Meng et al. (2021). We add two layers of Self-Attention and SOP classification head on top of the fused output for each slot. We train this model on Tesla GPUs.

7 Results

7.1 Joint Goal Accuracy (Overall Results)

We have achieved joint goal accuracy comparable to SOTA joint goal accuracy on MultiWOZ 2.2.

Model	Accuracy	Size
DS-DST (Zhang et al. (2019))	51.70	~440MB
SOM-DST baseline (Kim et al. (2020))	52	432MB
Gated-Self Attention DST (BERT-medium)	**53.30**	**202MB**
Gated-Self Attention DST (BERT-base)	**54.09**	**496MB**
Pegasus (Zhao et al. (2021))	56.60	>2.2GB
T5 (Zhao et al. (2021))	57.60	>891MB

Table 2: Joint Goal Accuracy on MultiWOZ 2.2

7.2 SOP Efficiency

F1 Score for State Operation Prediction (SOP) module.

Model	Operation			
	Delete	Update	Don't-Care	Carry-Over
SOM-DST (Baseline)	22.05	91.56	54.67	99.60
Intent Prediction (Joint Loss Optimization)	14.41	91.55	55.41	99.60
Appending intent logits-to SOP module	22.41	91.81	55.16	99.61
Intent and Slot-Self Attention	22.05	91.66	58.82	99.61
Gated-Intent (proposed model)	20.16	91.89	58.80	99.62

Table 3: SOP scores (F1) for each operation for dialog state borrowed from GT

8 Analysis

In Table 2, we compare our system's performance with other important works. Our Gated-Self Attention based model achieves a Joint Goal Accuracy (JGA) of 54.09 using BERT-base, and of 53.30 using BERT-medium. The system performance is comparable with the current SOTA results, while also providing the benefit of lesser processing. We achieve a performance improvement of 1.3% JGA, over the SOM-DST baseline.

We also compare the sizes of the different pre-trained models used by different systems, which gives a hint of the comparative memory efficiency of the models. Compared to T5 and Pegasus, our model makes use of BERT-medium , which is 4 times and 10 times smaller, respectively. Our model size is 202 MB which makes it feasible to deploy on-device.

As presented in Table 3, we also observe significant improvements in SOP efficiency, indicating optimization of the calls made to the time-consuming Slot Value Generator (SVG) module, further decreasing the overall latency of the system. The improvements are consistent across all the state-operations (Delete, Update, Don't Care, and Carry Over).

8.1 Future scope of enhancements

Similar improvisation can further be extended to dialog acts, which are more generic than intents, for SOP tasks.

We also plan to explore quantization techniques for reducing the model size without affecting the prediction results.

Another technique that has shown benefits in the task of named entity recognition is the use of external knowledge bases, for ever-expanding dynamic entities. We can further improve our system by incorporating such knowledge.

A limitation of our current system is the upper cap on the length of input (512 tokens). We would like to explore techniques to handle longer input sequences.

9 Conclusion

From our experiment results, we can conclude that using gating based self-attention on the intent logits for state operation prediction improves the accuracy. There is also a significant reduction in model size and latency when compared to other existing SOTA models which use large pre-trained language models. This makes our model more suitable for on-device based production environment.

References

Sungdong Kim, Sohee Yang, Gyuwan Kim, and Sang-Woo Lee. 2020. Efficient Dialogue State Tracking by Selectively Overwriting Memory. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 567–582, Online. Association for Computational Linguistics.

Rastogi, A., Zang, X., Sunkara, S., Gupta, R., & Khaitan, P. (2020). Towards Scalable Multi-Domain Conversational Agents: The Schema-Guided Dialogue Dataset. Proceedings of the AAAI Conference on Artificial Intelligence, 34(05), 8689-8696. https://doi.org/10.1609/aaai.v34i05.6394

Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, 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, Online. Association for Computational Linguistics.

Yann N. Dauphin, Angela Fan, Michael Auli, David Grangier. 2017. Language Modeling with Gated Convolutional Networks. Proceedings of the 34th International Conference on Machine Learning, in Proceedings of Machine Learning Research 70:933-941 Available from https://proceedings.mlr.press/v70/dauphin17a.html

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable Multi-Domain State Generator for Task-Oriented Dialogue Systems. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 808–819, Florence, Italy. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1):1929–1958.

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.

Diederik P Kingma and Jimmy Ba. 2015. *Adam: A method for stochastic optimization. In ICLR*.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Jeffrey Zhao, Mahdis Mahdieh, Ye Zhang, Yuan Cao, and Yonghui Wu. 2021. Effective Sequence-to-Sequence Dialogue State Tracking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7486–7493, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Chia-Hsuan Lee, Hao Cheng, and Mari Ostendorf. 2021. Dialogue State Tracking with a Language Model using Schema-Driven Prompting. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 4937–4949, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xin Tian, Liankai Huang, Yingzhan Lin, Siqi Bao, Huang He, Yunyi Yang, Hua Wu, Fan Wang, and Shuqi Sun. 2021. Amendable Generation for Dialogue State Tracking. In Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI, pages 80–92, Online. Association for Computational Linguistics.

Chen, Chun-Fu Richard, Quanfu Fan, and Rameswar Panda. "Crossvit: Cross-attention multi-scale vision transformer for image classification." In Proceedings of the IEEE/CVF international conference on computer vision, pp. 357-366. 2021.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." Advances in neural information processing systems 30 (2017).

BERT-Medium: https://huggingface.co/google/bert_uncased_L-8_H-512_A-8

Meng, Tao, Anjie Fang, Oleg Rokhlenko, and Shervin Malmasi. "GEMNET: Effective gated gazetteer representations for recognizing complex entities in low-context input." In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*

V. Agarwal, S. D. Shivnikar, S. Ghosh, H. Arora and Y. Saini, "LIDSNet: A Lightweight on-device Intent Detection model using Deep Siamese Network," 2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA), 2021, pp. 1112-1117, doi: 10.1109/ICMLA52953.2021.00182.

S. Ghosh, S. V. Gothe, C. Sanchi, and B. R. K. Raja, “edATLAS: An Efficient Disambiguation Algorithm for Texting in Languages with Abugida Scripts,” in 2021 IEEE 15th International Conference on Semantic Computing (ICSC), Jan 2021, pp. 325–332.

V. Agarwal, S. Ghosh, K. Ch, B. Challa, S. Kumari, Harshavardhana, and B. R. Kandur Raja, “EmpLite: A lightweight sequence labeling model for emphasis selection of short texts,” in Proceedings of the Workshop on Joint NLP Modelling for Conversational AI @ ICON 2020. Patna, India: NLP Association of India (NLPAI), Dec. 2020, pp. 19–26.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, and Dilek Hakkani Tür. 2019. Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines.

Raffel, Colin, et al. "Exploring the limits of transfer learning with a unified text-to-text transformer." J. Mach. Learn. Res. 21.140 (2020): 1-67.

Zhang, J.G., Hashimoto, K., Wu, C.S., Wan, Y., Yu, P.S., Socher, R. and Xiong, C., 2019. Find or classify? dual strategy for slot-value predictions on multi-domain dialog state tracking. *arXiv preprint arXiv:1910.03544*.

Emotion-guided Cross-domain Fake News Detection using Adversarial Domain Adaptation

Arjun Choudhry[†*], **Inder Khatri**[†*], **Arkajyoti Chakraborty**[†],
Dinesh Kumar Vishwakarma[†], **Mukesh Prasad**[‡]
[†]Biometric Research Laboratory, Delhi Technological University, New Delhi, India
[‡]School of Computer Science, University of Technology Sydney, Australia

Abstract

Recent works on fake news detection have shown the efficacy of using emotions as a feature or emotions-based features for improved performance. However, the impact of these emotion-guided features for fake news detection in cross-domain settings, where we face the problem of domain shift, is still largely unexplored. In this work, we evaluate the impact of emotion-guided features for cross-domain fake news detection, and further propose an emotion-guided, domain-adaptive approach using adversarial learning. We prove the efficacy of emotion-guided models in cross-domain settings for various combinations of source and target datasets from FakeNewsAMT, Celeb, Politifact and Gossipcop datasets.

1 Introduction

In recent years, our reliance on social media as a source of information has increased multi-fold, bringing along exponential increase in the spread of *fake news*. To counter this, researchers have proposed various approaches for fake news detection (Shu et al., 2019; Sheng et al., 2022). However, models trained on one domain are often brittle and vulnerable to incorrect predictions for the samples of another domain (Saikh et al., 2019; Pérez-Rosas et al., 2018). This is primarily due to the shift between the two domains, as depicted in Figure 1(1). To handle this, some domain-adaptive frameworks (Zhang et al., 2020; Huang et al., 2021; Li et al., 2021) have been proposed which help align the source and target domains in the feature space to ameliorate domain shift across different problems. These frameworks guide the feature extractors to extract domain-invariant features by aligning the source and target domains in the feature space, thus generalizing well across domains. However, due to the absence of labels in the target-domain data, the adaptation is often prone to negative transfer, which can disturb the class-wise distribution and affect the discriminability of the final model, as shown in Figure 1(2).

Some recent studies have observed a correlation between the veracity of a text and its emotions. There exists a prominent affiliation for certain emotions with fake news, and for other emotions with real news (Vosoughi et al., 2018), as illustrated in Figure 1(3). Further, some works have successfully utilized emotions as features, or emotion-guided features to aid in fake news detection (Guo et al., 2019; Zhang et al., 2021; Choudhry et al., 2022). However, we observe that these works only consider the in-domain setting for evaluation, without analyzing the robustness of these frameworks to domain shift in cross-domain settings. This is another important direction that needs to be explored.

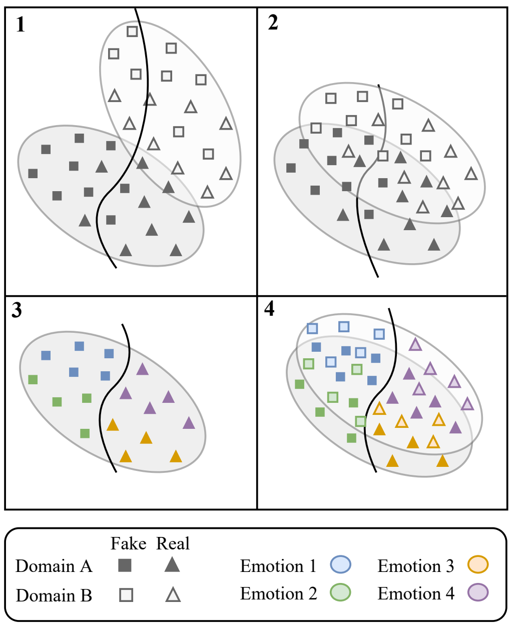


Figure 1: (1) Cross-domain texts not aligned. (2) Domain adaptation leads to some alignment. (3) Emotion-guided classification in one domain. (4) Emotion-guided domain adaptation leads to improved alignment of the two domains.

*Equal contribution

In this paper, we study the efficacy of emotion-aided models in capturing better generalizable features for cross-domain fake news detection. Table 1 shows the improvements observed in various cross-domain settings when our emotion-guided models were evaluated in cross-domain settings. We observe that emotion-guided frameworks show improved performance in cross-domain settings, as compared to their baseline models without the said emotion-aided features, thus underscoring the generalized feature extraction in emotion-aided models. We further propose an emotion-guided unsupervised domain adaptation framework, which utilizes emotion labels in a multi-task adversarial setting for better feature alignment across domains. The emotion labels for emotion classification, trained parallel to the fake news detection branch in the multi-task learning setup, help provide additional supervision for improved alignment during domain adaptation, mitigating the issue of incorrect alignment of domains. This is illustrated in Figure 1(4)). This leads to better discriminability. We experimentally prove the efficacy of our approach across a variety of datasets in cross-domain settings for various combinations of single-task or multi-task, domain-adaptive or non-adaptive, emotion-guided or unguided settings on the accuracy of the models.

Our contributions can be summarized as follows:

- We suggest the use of emotion classification as an auxiliary task for improved fake news detection in cross-domain settings, indicating the applicability of emotion-guided features across domains.

- We compare how Ekman's and Plutchik's base emotion classes individually affect the performance of our multi-task domain-adaptive framework, and if there are meaningful differences between them.

- We propose an emotion-guided domain-adaptive framework for fake news detection across domains. We show that domain-adaptive fake news detection models better align the two domains with the help of supervised learning using emotion-aided features.

- We evaluate our approach on a variety of source and target combinations from four datasets. Our results prove the efficacy of our approach.

2 Related Works

Several studies over the last few years have explored the correlation of fake news detection with emotions. K et al. (2020) *emotionized* text representations using explicit emotion intensity lexicons. Guo et al. (2019) utilized the discrepancies between publisher's emotion and the thread's comments' emotions to detect fake news. However, most of these methods relied upon some additional inputs during evaluation. Choudhry et al. (2022) proposed an emotion-aided multi-task learning approach, where emotion classification was the auxiliary task implicitly aligning fake news features according to emotion labels.

Inspired by Ganin et al. (2015), Zhang et al. (2020) proposed the first fake news detection work to tackle domain shifts between different datasets. They proposed a multi-modal framework with a Gradient Reversal Layer (GRL) to learn domain-invariant features across different domains and used a joint fake news detector on the extracted features. Huang et al. (2021) proposed a robust and generalized fake news detection framework adaptable to a new target domain using adversarial training to make the model robust to outliers and Maximum Mean Difference (MMD)-based loss to align the features of source and target. Li et al. (2021) extended the problem by treating it as a multi-source domain adaptation task, using the labeled samples from multiple source domains to improve the performance on unlabeled target domains. They also utilized weak labels for weak supervision on target samples to improve performance.

However, no previous work has aligned features between different domains using emotion-guided features and domain adaptation using adversarial training. We show that applying both of these approaches leads to improved performance due to better alignment of inter-domain features.

3 Proposed Methodology

3.1 Datasets, Emotion Annotation & Preprocessing

We use the FakeNewsAMT (Pérez-Rosas et al., 2018), Celeb (Pérez-Rosas et al., 2018), Politifact[1], and Gossipcop[2] datasets for cross-domain fake news detection. FakeNewsAMT is a multi-domain dataset containing samples from technology, education, business, sports, politics, and en-

[1] https://www.politifact.com [2] https://www.gossipcop.com

tertainment domains. The Celeb dataset has been derived from the web, and contains news about celebrities. Politifact is a web-scrapped dataset containing political news, while Gossipcop contains news extracted from the web, manually annotated via crowd-sourcing and by experts.

We use the Unison model (Colnerič and Demšar, 2020) to annotate all datasets with the core emotions from Ekman's (Ekman, 1992) (6 emotions: *Joy*, *Surprise*, *Anger*, *Sadness*, *Disgust*, *Fear*) and Plutchik's (Plutchik, 1982) (8 emotions: *Joy*, *Surprise*, *Trust*, *Anger*, *Anticipation*, *Sadness*, *Disgust*, *Fear*) emotion theories. During preprocessing, we convert text to lowercase, remove punctuation, and de-contract verb forms (eg. *"I'd"* to *"I would"*).

3.2 Multi-task Learning

We use multi-task learning (MTL) to incorporate emotion classification as an auxiliary task to our fake news detection branch. Multi-task learning enables a model to learn the shared features between two or more correlated tasks for improved feature extraction and performance. We use Ekman's or Plutchik's emotions labels for emotion classification branch in our MTL models to see which performs better, and compare the performance with the corresponding single-task (STL) models in domain-adaptive and non-adaptive settings.

3.3 Emotion-guided Domain-adaptive Framework

We propose the cumulative use of domain adaptation and emotion-guided feature extraction for cross-domain fake news detection. Our approach aims to improve the feature alignment between different domains using adversarial domain adaptation, by leveraging the correlation between the emotion and the veracity of a text (as shown in Figure 1(4)). Figure 2 shows our proposed framework. We use an LSTM-based (Hochreiter and Schmidhuber, 1997) feature extractor, which is trained using the accumulated loss from fake news classifier, emotion classifier and the discriminator (aids in learning domain-invariant features). LSTM can be replaced with better feature extractors. We used it specifically for easier comparison to non-adapted emotion-guided and non-adapted single-task models. The domain classifier acts as the discriminator. In our proposed framework, we do not use the truth labels for the target dataset for domain adaptation. However, we utilize the target domain emotion labels in our approach to better align the two domains using the emotion labels for supervised learning. The fake news classification loss, emotion classification loss, adversarial loss, and total loss are defined as in Equations 1, 2, 3, and 4:

$$L_{FND} = \min_{\theta_l, \theta_f} \sum_{i=1}^{n_s} L_f^i \tag{1}$$

$$L_{emo} = \min_{\theta_l, \theta_e} \sum_{i=1}^{n_s} L_{es}^i + \sum_{j=1}^{n_t} L_{et}^j)) \tag{2}$$

$$L_{adv} = \min_{\theta_d}(\max_{\theta_l}(\sum_{i=1}^{n_s} L_{ds}^i + \sum_{j=1}^{n_t} L_{dt}^j)) \tag{3}$$

$$L_{Total} = (1-\alpha-\beta)*L_{FND} + \alpha*(L_{adv}) + \beta*(L_{emo}) \tag{4}$$

where n_s and n_t are number of samples in source and target sets; θ_d, θ_f, θ_e and θ_l are parameters for discriminator, fake news classifier, emotion classifier and LSTM feature extractor; L_{d_s} and L_{d_t} are binary crossentropy loss for source and target classification; L_{es} and L_{et} are crossentropy loss for emotion classification; L_f is binary crossentropy loss for Fake News Classifier; α and β are weight parameters in L_{Total}. We optimised α and β for each setting for optimal performance.

We used 300 dimension GloVe (Pennington et al., 2014) embeddings. All models were trained for up to 50 epochs, stopped when the peak validation accuracy for the in-domain validation set was achieved. We used a batch size of 25 for every experiment. Each model used the Adam optimizer with learning rate 0.0025. We used an LSTM layer with 256 units for feature extraction, while both fake news detection and emotion classification branches consisted of two dense layers each.

4 Experimental Analysis & Results

We evaluated our proposed approach on various combinations of source and target datasets. Each model was optimized on an in-domain validation set from the source set. Table 1 illustrates our results proving the efficacy of using emotion-guided models in non-adapted and domain-adapted cross-domain settings. It compares non-adaptive models, domain-adaptive models, and our emotion-guided domain-adaptive models in various settings. MTL(E) and MTL(P) refer to emotion-guided multi-task frameworks using Ekman's and Plutchik's emotions respectively. STL refers to the single-task framework. DA refers to the use of the domain-adaptive framework, containing a discriminator. Some findings observed are:

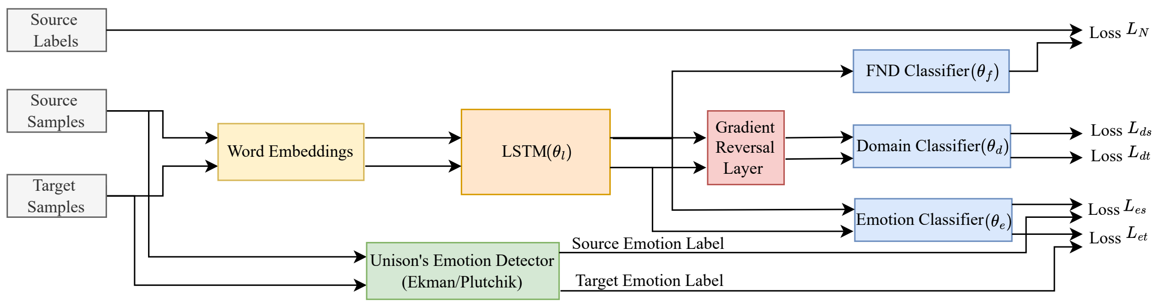


Figure 2: Pictorial depiction of our emotion-guided domain-adaptive approach for cross-domain fake news detection.

Source	Target	Setting	Accuracy
FakeNewsAMT	Celeb	STL	0.420
		MTL(E)	0.520
		MTL(P)	0.530
		DA STL	0.560
		DA MTL(E)	0.540
		DA MTL(P)	**0.600**
Celeb	FakeNewsAMT	STL	0.432
		MTL(E)	0.471
		MTL(P)	0.476
		DA STL	0.395
		DA MTL(E)	0.501
		DA MTL(P)	**0.551**
Politifact	Gossipcop	STL	0.527
		MTL(E)	0.555
		MTL(P)	0.603
		DA STL	0.585
		DA MTL(E)	**0.698**
		DA MTL(P)	0.671
Celeb	Gossipcop	STL	0.488
		MTL(E)	0.501
		MTL(P)	0.490
		DA STL	0.525
		DA MTL(E)	0.555
		DA MTL(P)	**0.587**
FakeNewsAMT	Gossipcop	STL	0.451
		MTL(E)	0.652
		MTL(P)	0.620
		DA STL	0.790
		DA MTL(E)	**0.805**
		DA MTL(P)	0.795
FakeNewsAMT	Politifact	STL	0.363
		MTL(E)	0.450
		MTL(P)	0.530
		DA STL	0.621
		DA MTL(E)	**0.704**
		DA MTL(P)	0.621

Table 1: Cross-domain evaluation of non-adaptive and adaptive models on FakeNewsAMT, Celeb, Politifact and Gossipcop datasets. Emotion-guided domain-adaptive models (DA MTL(E) and DA MTL(P)) outperform their corresponding STL models in cross-domain settings. Domain-adaptive MTL models consistently outperform baseline STL, non-adaptive MTL and domain-adaptive STL models.

MTL(E) and MTL(P) models outperform their STL counterparts in cross-domain settings, as seen in Table 1. This indicates improved extraction of generalizable features by the emotion-guided models, which aids in improved fake news detection across datasets from different domains.

DA STL models generally outperform STL models in cross-domain settings across multiple combinations of datasets. However, we see the STL model outperformed the DA STL model for Celeb dataset as the source dataset and FakeNewsAMT dataset as target, confirming that unguided adaptation can sometimes lead to negative alignment, reducing the performance of the model.

DA MTL(E) and DA MTL(P) models improve performance in cross-domain settings. Table 1 shows improved results obtained using the emotion-guided adversarial DA models over their non-adaptive counterparts. This shows the scope for improved feature extraction even after using DA, and emotion-guided models act as a solution aiding in correct alignment of the samples and features extracted by the adaptive framework from different domains. Emotion-guided DA models mitigated the issue of negative alignment when Celeb dataset was the source and FakeNewsAMT dataset the target, where STL model outperformed the DA STL model. The emotion-guided DA models helped correctly align the two domains, leading to significantly improved performance.

5 Conclusion

In this work, we showed the efficacy of emotion-guided models for improved cross-domain fake news detection, and presented an emotion-guided domain-adaptive fake news detection approach, evaluating it against baseline STL, emotion-guided MTL, DA STL and emotion-guided DA MTL models for various source and target combinations from four datasets. Our proposed approach led to improved cross-domain fake news detection accuracy, indicating that emotions are generalizable across domains and aid in better alignment of different domains during domain adaptation.

References

Arjun Choudhry, Inder Khatri, and Minni Jain. 2022. An emotion-based multi-task approach to fake news detection (student abstract). *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(11):12929–12930.

Niko Colnerič and Janez Demšar. 2020. Emotion recognition on twitter: Comparative study and training a unison model. *IEEE Transactions on Affective Computing*, 11(3):433–446.

P. Ekman. 1992. An argument for basic emotions. *Cognition & Emotion*, 6.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2015. Domain-adversarial training of neural networks.

Chuan Guo, Juan Cao, Xueyao Zhang, Kai Shu, and Miao Yu. 2019. Exploiting emotions for fake news detection on social media. *ArXiv*, abs/1903.01728.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Yinqiu Huang, Min Gao, Jia Wang, and Kai Shu. 2021. DAFD: domain adaptation framework for fake news detection. In *Neural Information Processing - 28th International Conference, ICONIP 2021, Sanur, Bali, Indonesia, December 8-12, 2021, Proceedings, Part I*, volume 13108 of *Lecture Notes in Computer Science*, pages 305–316. Springer.

Anoop K, Deepak P, and Lajish V L. 2020. Emotion cognizance improves health fake news identification. In *Proceedings of the 24th Symposium on International Database Engineering & Applications*, IDEAS '20. Association for Computing Machinery.

Yichuan Li, Kyumin Lee, Nima Kordzadeh, Brenton Faber, Cameron Fiddes, Elaine Chen, and Kai Shu. 2021. Multi-source domain adaptation with weak supervision for early fake news detection. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 668–676.

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.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2018. Automatic detection of fake news. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3391–3401. Association for Computational Linguistics.

Robert Plutchik. 1982. A psychoevolutionary theory of emotions. *Social Science Information*, 21(4-5).

Tanik Saikh, Arkadipta De, Asif Ekbal, and Pushpak Bhattacharyya. 2019. A deep learning approach for automatic detection of fake news. In *Proceedings of the 16th International Conference on Natural Language Processing*, pages 230–238, International Institute of Information Technology, Hyderabad, India. NLP Association of India.

Qiang Sheng, Juan Cao, Xueyao Zhang, Rundong Li, Danding Wang, and Yongchun Zhu. 2022. Zoom out and observe: News environment perception for fake news detection.

Kai Shu, Limeng Cui, Suhang Wang, Dongwon Lee, and Huan Liu. 2019. Defend: Explainable fake news detection. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 395–405, New York, NY, USA. Association for Computing Machinery.

Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. *Science*, 359(6380):1146–1151.

Tong Zhang, Di Wang, Huanhuan Chen, Zhiwei Zeng, Wei Guo, Chunyan Miao, and Lizhen Cui. 2020. Bdann: Bert-based domain adaptation neural network for multi-modal fake news detection. In *IJCNN*.

Xueyao Zhang, Juan Cao, Xirong Li, Qiang Sheng, Lei Zhong, and Kai Shu. 2021. Mining dual emotion for fake news detection. In *Proceedings of the Web Conference 2021*, WWW '21, page 3465–3476, New York, NY, USA. Association for Computing Machinery.

Generalised Spherical Text Embedding

Souvik Banerjee[1], **Bamdev Mishra**[2], **Pratik Jawanpuria**[2], and **Manish Shrivastava**[1]
[1] International Institute of Information Technology, Hyderabad
[2] Microsoft India
souvik.banerjee@research.iiit.ac.in
bamdevm@microsoft.com
pratik.jawanpuria@microsoft.com
m.shrivastava@iiit.ac.in

Abstract

This paper aims to provide an unsupervised modelling approach that allows for a more flexible representation of text embeddings. It jointly encodes the words and the paragraphs as individual matrices of arbitrary column dimension with unit Frobenius norm. The representation is also linguistically motivated with the introduction of a novel similarity metric. The proposed modelling and the novel similarity metric exploits the matrix structure of embeddings. We then go on to show that the same matrices can be reshaped into vectors of unit norm and transform our problem into an optimization problem over the spherical manifold. We exploit manifold optimization to efficiently train the matrix embeddings. We also quantitatively verify the quality of our text embeddings by showing that they demonstrate improved results in document classification, document clustering, and semantic textual similarity benchmark tests.[1].

1 Introduction

Most unsupervised text embedding models are trained by encoding the words or paragraphs acquired from the training data as a feature length vector, with the assumption that they reside in the Euclidean space. Such models are ubiquitous for good reason. Aside from their efficiency, they have also proven to be very effective providing us with state of the art results in various intrinsic and extrinsic embedding evaluation tasks. Word2vec(Mikolov et al., 2013b,a), and GLoVE (Pennington et al., 2014) are two notable examples where word embeddings are learned in the Euclidean space and are trained to be oriented such that word vectors that appear in the same context have higher cosine similarity. Some of the most common methods of intrinsic evaluation of word embeddings include word similarity, word analogy, and compositionality. Doc2vec (Le and Mikolov, 2014), an unsupervised document embedding model generalises the training method introduced in Word2vec to documents and achieves improved results in various downstream tasks like sentiment analysis, information retrieval and multi-class classification. There are other document embedding models like skip-thought (Kiros et al., 2015) and infersent (Conneau et al., 2017; Moghadasi and Zhuang, 2020).

The joint spherical embedding model, JoSE as proposed in (Meng et al., 2019), shows that directional similarity is often more effective in tasks such as word similarity and document clustering. They show that when embeddings are trained in the Euclidean space, there is a performance gap between the training stage and usage stage of text embeddings. To bridge that gap, they propose a model which trains both words and paragraphs on a spherical space with tools from Riemannian optimization methods. The resulting embeddings are also shown to give considerably better results in word similarity, document clustering, and document classification tasks when compared with other standard models. Such application of manifold geometry has also been explored in substantial depth in works like (Batmanghelich et al., 2016; Reisinger et al., 2010; Gopal and Yang, 2014). There are also other notable Riemannian optimization based embedding training models like (Tifrea et al., 2018; Nickel and Kiela, 2017) which train embeddings on the hyperbolic manifold space and uses its tree like property for better hierarchical representation of data. Hyperbolic word embeddings are also intrinsically linked with Gaussian word embeddings (Vilnis and McCallum, 2014) which gives a lot more insight into the geometry of word embeddings.

However, most of these text embedding models like JoSE, Word2vec, Doc2vec, and fastText (Bojanowski et al., 2017; Meng et al., 2020) are trained with the goal of getting a single dense vector representation per word or document. These

[1] https://github.com/SouvikBan/matrix_rep

models treat both polysemous and monosemous words in the same way resulting in the most frequent meaning of the word dominating the others or the meanings getting mixed in the case of former. It is especially detrimental for documents where we use a single dense vector representation to encode information which span over several sentences, often involving multiple topics.

This paper aims to address this problem by using matrices as the mode of representation instead of vectors. Our model is the joint word and document training generative model proposed in `JoSE` where we replace the cosine similarity metric with a novel metric that exploits the matrix structure of the embeddings. This robust metric takes word or document matrices of arbitrary number of columns and calculates the similarity between them. We also show that a few reshape operations allow us to reformulate the optimization problem of our model in terms of the spherical manifold optimization problem. Thus, we offer more flexibility in the way of matrix dimensions while retaining efficiency. Our choice of metric also suggests that the word, sentence, paragraph/document embeddings do not need to have the same number of columns, which has linguistic validation.

2 Matrix Representation of Texts and Optimization Problem

The text embeddings are represented as elements of the following set

$$\mathcal{S}(p,r) = \{\mathbf{X} \in \mathbb{R}^{p\times r} : ||\mathbf{X}||_F = 1\},$$

where $r \leq p$ and $\|\cdot\|_F$ denotes the Frobenius norm. The Frobenius norm is the matrix norm of a $p \times r$ matrix $\mathbf{X}$ defined as the square root of the sum of the absolute squares of its elements, i.e.,

$$||\mathbf{X}||_F = \sqrt{\sum_{i=1}^{p}\sum_{j=1}^{r} x_{ij}^2}.$$

Our model design is consistent with `JoSE` where it is assumed that text generation is a two-step process: a center word is first generated according to the semantics of the paragraph, and then the surrounding words are generated based on the center word's semantics. Consider a positive tuple $(\mathcal{U}, \mathcal{V}, \mathcal{D})$ where word $\mathcal{V}$ appears in the local context window of word $\mathcal{U}$ in paragraph $\mathcal{D}$ and negative tuple $(\mathcal{V}, \mathcal{U}', \mathcal{D})$ where $\mathcal{U}'$ is a randomly sampled word from the vocabulary serving as a negative sample. We represent words $\mathcal{V}, \mathcal{U}, \mathcal{U}'$ as matrices $\mathbf{V}, \mathbf{U}, \mathbf{N}$ which are elements of the set $\mathcal{S}(p, r_1)$ and paragraph $\mathcal{D}$ as matrix $\mathbf{D}$ which is an element of the set $\mathcal{S}(p, r_2)$, where $p, r_1, r_2 > 0$. From a linguistic perspective, these matrices can be considered as a set of latent variables that govern the semantics of a word or a document. Each column is given some arbitrary unit of linguistic information to encode, a latent variable which contributes to the mathematical representation of a word or a document. For example, the columns of a matrix $\mathbf{D}$ that represent the document $\mathcal{D}$ might encode latent variables that contain information about some topic contained in that document. Similarly, the columns of the word matrix $\mathbf{U}$ might encode information about a specific context in which a polysemous word $\mathcal{U}$ appears. We also keep the number of columns for word matrices less than or equal to the number of columns for sentence/document matrices, i.e., $r_1 \leq r_2$, so that the number of latent variables governing a word should not be more than the ones that govern a sentence or paragraph.

Novel metric. To model the above mentioned linguistic representation mathematically, we define a novel similarity metric for the ambient space in which we train our matrix embeddings. The proposed metric function is a measure of similarity between two sets of latent variables (matrices) - a function analogous to the cosine similarity measure for vectors in the Euclidean space. Given two arbitrary matrices $\mathbf{A} \in \mathcal{S}(p, r_1), \mathbf{B} \in \mathcal{S}(p, r_2)$, we propose the similarity metric $\mathbf{g} : \mathcal{S}(p, r_1) \times \mathcal{S}(p, r_2) \to \mathbb{R}$ as

$$\mathbf{g}(\mathbf{A}, \mathbf{B}) = \frac{\sum_{i=1}^{r_1}\sum_{j=1}^{r_2} a_i^\top b_j}{r_1 r_2}, \tag{1}$$

where $\mathbf{A} = [a_1\, a_2 \cdots a_{r_1}]$, $\mathbf{B} = [b_1\, b_2 \cdots b_{r_2}]$, $a_i, b_j \in \mathbb{R}^p\, \forall i \in [1, 2, \cdots r_1], \forall j \in [1, 2, \cdots r_2]$.

Motivation for our similarity metric. The metric $\mathbf{g}$ (1) calculates the average of all the entries in the matrix $\mathbf{A}^\top\mathbf{B}$. The linguistic intuition behind the choice of this metric is that we want to define a metric that takes the average of dot products between all possible pairs of latent variables (columns) from each matrix. In the case of $r_1 = r_2 = 1$, $\mathbf{g}(\mathbf{A}, \mathbf{B})$ reduces to the cosine similarity metric between unit norm vectors $\mathbf{A}$ and $\mathbf{B}$ which is the metric used in the spherical space model of `JoSE`. However, in the case of higher values of r_1, r_2, For example, let two words $\mathcal{V}_1$ and $\mathcal{V}_2$ be represented by proposed $p \times r_1$ matrix embeddings - $\mathbf{V}_1 = [\mathrm{a}_1, \mathrm{a}_2]$ and $\mathbf{V}_2 = [\mathrm{b}_1, \mathrm{b}_2]$, where $p = 1$ and $r_1 = 2$. The proposed similarity metric $\mathbf{g}(\mathbf{V}_1, \mathbf{V}_2)$ is computed as $(a_1b_1 + a_1b_2 + a_2b_1 + a_2b_2)/4$. Note that this is different from computing the cosine similarity which gives $(a_1b1 + a_2b_2)$. Moreover, the regular cosine similarity between word and paragraph embedding matrices with unequal dimensions ($p \times r_1$ and $p \times r_2$ respectively) is not defined. On the other hand, our proposed similarity metric is still applicable.

Modelling. As our model has the same generative process as `JoSE`, we take the same max-margin loss function and substitute the cosine similarity metric with our new similarity metric **g** where the word matrices $\mathbf{U}, \mathbf{V}, \mathbf{N} \in \mathcal{S}(p, r_1)$ and paragraph matrix $\mathbf{D} \in \mathcal{S}(p, r_2)$ with $r_1 \leq r_2$. We get the following loss, i.e.,

$$\begin{aligned}&\mathcal{L}(\mathbf{V}, \mathbf{U}, \mathbf{N}, \mathbf{D}) \\ &\quad = \max\left(0, m - \mathbf{g}(\mathbf{V}, \mathbf{U}) - \mathbf{g}(\mathbf{U}, \mathbf{D})\right. \\ &\qquad\qquad \left. + \mathbf{g}(\mathbf{V}, \mathbf{N}) + \mathbf{g}(\mathbf{N}, \mathbf{D})\right).\end{aligned} \tag{2}$$

where $m > 0$ is the margin.

Optimization. For the purpose of optimization, matrices of different dimensions are reshaped and embedded into Riemannian spherical manifolds of different dimensions. Overall, they are combined using the Riemannian product manifold structure. Therefore, the optimization of $\mathcal{L}$ (2) is done by performing two reshape operations per iteration while training. For example, the unit Frobenius norm matrices of dimension $\mathbb{R}^{p\times r}$ can be reshaped into vectors of dimension $\mathbb{R}^{pr}$ with the unit norm. To calculate the value of our loss function (2) at every iteration and the Euclidean gradient (partial derivatives), the vectors in question are reshaped into matrices for calculating the **g** values and their gradients. Subsequently, the matrices are reshaped back into vectors. We then apply the Riemannian gradient descent update rule to update the parameters (Meng et al., 2019; Absil et al., 2008; Smith, 2014; Edelman et al., 1998). Note that our proposed modelling and optimization are different from just training on the spherical manifold with unit vectors and using the cosine similarity metric (which is the case in `JoSE`).

3 Experiments

To highlight the quality of our obtained matrix representations, we run the same set of evaluations as `JoSE` with a relatively lower number of columns, i.e., $1 \leq r_1 \leq r_2 \leq 6$. We notice that for even higher values, the quality of our embeddings gradually decrease. Moreover, the word similarity experiment results are not added as words seemingly do not benefit from our representation directly. Indeed, the best word similarity score are obtained for $r_1 = r_2 = 1$. Instead, we add semantic textual similarity benchmark tests to show that sentences can benefit from this matrix representation model. Unless otherwise stated, our model and `JoSE` are trained for 35 iterations on the respective corpora; the local context window size is 5; the embedding dimension is kept at 100; the number of negative samples are 2. Other hyperparameters in our model are kept the same as `JoSE`.

3.1 Document Clustering

We perform document clustering on the 20 Newsgroup[2] dataset using spectral clustering. Each paragraph in the dataset is separated by a new line and is considered a separate document while training. `JoSE` uses K-Means and SK-Means as the clustering algorithm that assume the ambient space to be the Euclidean and the spherical space, respectively. Our non-Euclidean space with its custom metric requires a clustering algorithm that allows the freedom of using custom metric, i.e., the algorithm should be space agnostic. We found spectral clustering to suit those requirements perfectly. The four external measures used for validating the results are kept unchanged from `JoSE` (Banerjee et al., 2005; Manning et al., 2008; Steinley, 2004). These measures are Mutual Information (MI), Normalized Mutual Information (NMI), Adjusted Rand Index (ARI), and Purity. We run the clustering algorithm with our custom similarity metric as written in (1) with kernel coefficient, $\gamma = 0.001$. Table 1 shows quantitatively how matrix representations benefit document embeddings for clustering tasks. Keeping $r_1 = 1$ fixed, we see a steady increase in performance as r_2 is increased from 1 (the score of our baseline model - `JoSE`) to 6.

3.2 Document Classification

Following (Meng et al., 2019), the document classification evaluations are ran on the following two datasets: the topic classification 20 Newsgroup dataset (which we used for document clustering as well) and a binary sentiment classification dataset consisting of 1 000 positive and 1 000 negative movie reviews[3]. The train/test split is the origi-

[2] `http://qwone.com/~jason/20Newsgroups/`

[3] `http://www.cs.cornell.edu/people/pabo/movie-review-data/`

Table 1: Evaluation results for spectral clustering of document embeddings on the 20 Newsgroup dataset for kernel coefficient, $\gamma = 0.001$ ($r_1 = 1, r_2 = 1$ is JoSE score). Document embeddings benefit from matrix representations as demonstrated by better scores for higher values of r_2. Here, $r_1 = 1$.

$r_2, r_1 = 1$	MI	NMI	ARI	Purity
$r_2 = 1$	1.73	0.58	0.45	0.64
$r_2 = 2$	1.75	0.59	0.46	0.63
$r_2 = 3$	1.75	0.59	0.46	0.62
$r_2 = 4$	1.77	0.60	0.46	0.63
$r_2 = 5$	1.84	0.62	0.49	0.65
$r_2 = 6$	**1.85**	**0.62**	**0.49**	**0.67**

Table 2: F1-macro, F1-micro for 20 Newsgroup dataset classification using K-NN with K=3 ($r_1 = 1, r_2 = 1$ is JoSE score). Increasing the value r_2 benefits documents embeddings in classification tasks.

$r_1 \backslash r_2$	r_2=1	r_2=2	r_2=3	r_2=4
r_1=1	0.74, 0.74	0.77, 0.77	0.78, 0.78	**0.78, 0.78**
r_1=2	–	0.76, 0.76	0.77, 0.77	0.76, 0.77
r_1=3	–	–	0.76, 0.76	0.73, 0.74
r_1=4	–	–	–	0.72, 0.72

nal split for 20 Newsgroup while for the movie review datasets, the splitting is done by randomly selecting 80% of the data as training and 20% as testing. The classification algorithm we use is K-NN with $k = 3$ and a custom distance metric that is suitable for our space. The custom distance metric for two paragraph matrices $\mathbf{U} = [u_1\, u_2 \cdots u_{r_2}]$ and $\mathbf{V} = [v_1\, v_2 \cdots v_{r_2}]$ where $\mathbf{U}, \mathbf{V} \in \mathcal{S}(p, r_2)$ and $u_i, v_i \in \mathbb{R}^p\, \forall i \in [1, 2, \cdots, r_2]$ is defined as

$$\text{dist}^2(\mathbf{U}, \mathbf{V}) = \frac{\sum_{k=1}^{r_2} \sum_{l=1}^{r_2} (u_k - v_l)^\top (u_k - v_l)}{r_2^2}. \quad (3)$$

The intuition for the distance metric in (3) comes from our interpretation of each individual column as encoding a latent variable governing the semantics of that specific document. A quick look at (3) tells us that the distance metric takes the square root of the average of the squared Euclidean distances between all pairs of columns formed from one matrix with another. Tables 2 and 3 list the Macro-F1 and Micro-F1 scores for 20 Newsgroup dataset and Movie Reviews dataset respectively for increasing values of r_1 and r_2. We again see an increase in scores for higher values of both r_2 and r_1 compared to JoSE (r_1=1, r_2=1).

3.3 Semantic Textual Similarity Task

Semantic Textual Similarity Benchmark comprises a selection of the English datasets used in the STS

Table 3: F1-macro, F1-micro for movie review dataset classification using K-NN with K=3 ($r_1 = 1, r_2 = 1$ is the JoSE score).

$r_1 \backslash r_2$	r_2=1	r_2=2	r_2=3	r_2=4
r_1=1	0.74, 0.74	0.75, 0.75	0.76, 0.76	0.75, 0.76
r_1=2	–	0.75, 0.75	0.75, 0.75	0.74, 0.74
r_1=3	–	–	0.74, 0.74	**0.76, 0.76**
r_1=4	–	–	–	0.74, 0.74

Table 4: Pearson Correlation for STS Benchmark on dev and test data ($r_1 = 1, r_2 = 1$ is the JoSE score). Even sentences can benefit from our matrix representation as demonstrated by better scores with higher values of r_2.

$r_1 \backslash r_2$	r_2=1	r_2=2	r_2=3	r_2=4
r_1=1	0.51, 0.40	0.51, 0.39	0.52, 0.40	0.53, 0.40
r_1=2	–	0.53, 0.40	0.53, 0.40	0.53, 0.40
r_1=3	–	–	0.53, 0.40	**0.54, 0.40**
r_1=4	–	–	–	0.53, 0.40

tasks organized in the context of SemEval (Cer et al., 2017) between 2012 and 2017[4]. We perform semantic textual similarity tasks on the sts-benchmark dataset to show that even sentences can benefit from being represented as matrices. The benchmark comprises of 8 628 sentence pairs split into 3 partitions: train, development and test. The results are reported on both the test and dev sets. Each sentence in the dataset is treated as a separate document by our model and we use all the sentences in the train, development and test set to train. The rationale for this is that the model is completely unsupervised, i.e., it takes only the raw text and uses no supervised or annotated information, and thus there is no need to hold out the test data as it is unlabelled. We train for 1 000 iters with window size 15 and negative samples 5 while the rest of the hyperparameters were kept at their default values. To score a sentence pair representation, similarity was computed between them using our custom metric described in 1 for our model. We report the dev and test Pearson correlation score for $r_1, r_2 = 1, 2, 3, 4,\ r_1 \leq r_2$. As Table 4 reports, higher values of r_2 give better scores compared to our baseline model JoSE ($r_1 = r_2 = 1$).

4 Conclusion

In this paper, we extend the joint modelling idea used for training text embeddings from vectors with unit norm to matrices with unit Frobenius norm.

[4]http://ixa2.si.ehu.eus/stswiki/index.php/STSbenchmark

Each word/sentence/document matrix is made to encode information in a way that each column of the matrix represents some latent topic, context, or discourse. Since the standard vector dot product can no longer be applied, we introduce a novel similarity metric that allows the measurement of similarity between matrices of arbitrary number of columns. For optimization simplicity, we reshape our matrices to vectors of unit norm that allows using the Riemannian gradient descent optimization algorithm on the spherical manifold. Our theory is validated quantitatively by the results which shows that our text embeddings outperform or produce similar results when compared with `JoSE` in document classification, clustering, and semantic textual similarity tasks.

This paper is meant to serve as a ground work for more involved research topics which integrate concepts of differential geometry and NLP. Future directions could include qualitative analysis on the columns of the matrices to see what tangible information they encode which will allow for better modelling. Another direction would be to exploit other matrix structures on the embeddings, e.g., treating each word embedding as a symmetric positive definite matrix and to study whether they can be beneficial.

References

P.-A. Absil, R. Mahony, and R. Sepulchre. 2008. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ.

Arindam Banerjee, Inderjit S. Dhillon, Joydeep Ghosh, and Suvrit Sra. 2005. Clustering on the unit hypersphere using von mises-fisher distributions. *J. Mach. Learn. Res.*, 6:1345–1382.

Kayhan Batmanghelich, Ardavan Saeedi, Karthik Narasimhan, and Sam Gershman. 2016. Nonparametric spherical topic modeling with word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 537–542, Berlin, Germany. Association for Computational Linguistics.

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.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo 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)*. Association for Computational Linguistics.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data.

Alan Edelman, Tomás A. Arias, and Steven T. Smith. 1998. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353.

Siddharth Gopal and Yiming Yang. 2014. Von mises-fisher clustering models. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 154–162, Bejing, China. PMLR.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, USA.

Yu Meng, Jiaxin Huang, Guangyuan Wang, Chao Zhang, Honglei Zhuang, Lance Kaplan, and Jiawei Han. 2019. Spherical text embedding. In *Advances in neural information processing systems*.

Yu Meng, Yunyi Zhang, Jiaxin Huang, Yu Zhang, Chao Zhang, and Jiawei Han. 2020. Hierarchical topic mining via joint spherical tree and text embedding. KDD '20, page 1908–1917, New York, NY, USA. Association for Computing Machinery.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mahdi Naser Moghadasi and Yu Zhuang. 2020. Sent2vec: A new sentence embedding representation with sentimental semantic. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 4672–4680.

Maximilian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations.

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.

Joseph Reisinger, Austin Waters, Bryan Silverthorn, and Raymond J. Mooney. 2010. Spherical topic models. ICML'10, page 903–910, Madison, WI, USA. Omnipress.

Steven Thomas Smith. 2014. Optimization techniques on riemannian manifolds.

Douglas L. Steinley. 2004. Properties of the hubert-arabie adjusted rand index. *Psychological methods*, 9 3:386–96.

Alexandru Tifrea, Gary Bécigneul, and Octavian-Eugen Ganea. 2018. Poincaré glove: Hyperbolic word embeddings.

Luke Vilnis and Andrew McCallum. 2014. Word representations via gaussian embedding.

CNN-Transformer based Encoder-Decoder Model for Nepali Image Captioning

Bipesh Subedi and **Bal Krishna Bal**
Information and Language Processing Research Lab
Kathmandu University, Dhulikhel, Nepal
bipeshrajsubedi@gmail.com
bal@ku.edu.np

Abstract

Many image captioning tasks have been carried out in recent years, the majority of the work being for the English language. A few research works have also been carried out for Hindi and Bengali languages in the domain. Unfortunately, not much research emphasis seems to be given to the Nepali language in this direction. Furthermore, the datasets are also not publicly available in the Nepali language. The aim of this research is to prepare a dataset with Nepali captions and develop a deep learning model based on the Convolutional Neural Network (CNN) and Transformer combined model to automatically generate image captions in the Nepali language. The dataset for this work is prepared by applying different data preprocessing techniques on the Flickr8k dataset. The preprocessed data is then passed to the CNN-Transformer model to generate image captions. ResNet-101 and EfficientNetB0 are the two pre-trained CNN models employed for this work. We have achieved some promising results which can be further improved in the future.

1 Introduction

Image Caption generation is one of the Computer Vision and Natural Language Processing (NLP) tasks which generates the description of an image. It can be used to aid visually impaired people for describing scenarios, for image indexing, creating image-based search engines, annotating news images, and many more. Automatically captioning an image is one of the main goals of scene understanding in Computer Vision. The caption generation model should handle the challenges of identifying the objects in an image as well as capturing and expressing their description in the natural language (Xu et al., 2015). It is a complex process compared to the existing object detection and classification tasks. In this regard, Computer Vision techniques are used to understand the contents and extract features from the image whereas the Natural Language Processing techniques generate words or descriptions from the extracted features in the right order (Srinivasan and Sreekanthan, 2018).

Recent advancements in NLP and Computer Vision have paved the way to perform various tasks using the native language. Image captioning for the Nepali language is one of the least researched topics and very limited literature are available owing to the language complexity and unavailability of datasets. The only work carried out in this domain was proposed by Adhikari and Ghimire (2019) which employs a traditional CNN-RNN encoder-decoder architecture. Hindi, Marathi, and Konkani are some of the other languages that share similar grammatical structures as the Nepali language as all of these languages use the Devanagari script. Additionally, Bengali language also belongs to the Indic language groups along with other languages mentioned above. Hence, this work is an attempt to develop image captioning for the Nepali language with reference to the existing works. Some of the existing works in this field include Hindi image captioning techniques shown by Mishra et al. (2021) and Bengali image captioning proposed by Palash et al. (2021).

In this research, we propose a CNN-Transformer-based Nepali Image Captioning model. The reason behind opting the transformer network is its wide applicability in today's Natural Language Processing domain as well as its performance in image captioning tasks for other languages. It is computationally faster than RNN as it supports parallel processing. Furthermore, transformer networks have not been implemented for Nepali image captioning so far and hence, this work to the best of our knowledge is the first to implement it. The main focus of this study is to create Nepali datasets and develop a CNN-Transformer model to generate Nepali captions. The remainder of this paper is arranged in the following order: Section 2 discusses some of the related works, Section 3 shows the methodology used, and Section 4 consists of results and discussions of the work followed by a conclusion and future directions.

2 Related Works

Various works have been carried out in the image captioning domain, especially in resourceful languages such as English. A significant number of research works can also be seen in Hindi and Bengali which are closely related to the Nepali language. Adhikari and Ghimire (2019) proposed the only work in Nepali that utilizes the two encoder-decoder models with and without visual attention inspired by (Xu et al., 2015). The models use ResNet-50 as encoder and LSTM/GRU as decoder respectively trained on MS COCO datasets after translating and preprocessing. Mishra et al. (2021) proposed a transformed-based encoder-decoder architecture for image captioning for the Hindi language where ResNet-101 is used as an encoder and Transformer as a decoder. They have outlined the problems with RNN architecture and proposed a transformer model with a stacked attention mechanism to decode the image feature vectors into sentences. The authors have calculated BLEU1, BLEU2, BLEU3, and BLEU4 scores with values of 62.9, 43.3, 29.1, and 19.0 respectively. Similarly, Palash et al. (2021), Shah et al. (2021), and Ami et al. (2020) have proposed image captioning in the Bengali language using CNN and transformer networks. Palash et al. (2021) have used ResNet-101 for extracting image features whereas Shah et al. (2021) and Ami et al. (2020) opted for two pre-trained CNN models: InceptionV3 and Xception. Palash et al. (2021) used BanglaLekha datasets for training and testing the model. Similarly, Ami et al. (2020) have used Flickr8k datasets after preprocessing and the work is extended by Shah et al. (2021) using BanglaLekha datasets. BLEU1, BLEU2, BLEU3, BLEU4, and METEOR scores obtained in Palash et al. (2021) work are 0.694, 0.580, 0.505, 2.22e-308, and 0.337 respectively which is better compared to (Mishra et al., 2021). Moreover, Shen et al. (2020) proposed a new model for remote sensing image captioning tasks for the English language. The transformer-based decoder was used to generate captions from the image features. The semantic and spatial features were extracted from the image using CNN. To capture a deeper relationship between image features and text descriptions, the semantic features were added to the decoder's every single sub-layer. The datasets used for this research were obtained from the Sydney Dataset, Remote Sensing Image Caption Dataset (RSICD), and UCM Dataset.

The above literature suggests that different works have been carried out in the image captioning domain in several languages and most of them have concluded that transformer networks perform better than traditional CNN-RNN architecture. In this regard, the Nepali image captioning system using transformers is yet to be explored and the Nepali caption datasets are also not publicly available. Based on these facts and literature, a CNN-Transformer model is attempted to implement in this research.

3 Methodology

The methodology employed for this research involves experimentation of data on the model architecture. The datasets for Nepali image captions are not available publicly, hence two procedures are followed for developing the Nepali image captioning system. The first step involves dataset preparation, followed by model architecture design and implementation.

3.1 Dataset Preparation

The dataset preparation involves two tasks: Dataset collection and Dataset preprocessing. The dataset thus prepared is divided into three parts - for training, validation, and testing purposes.

3.1.1 Dataset Collection

The datasets used for this research are collected from the Flickr8k public dataset[1]. It consists of more than 8000 images with 5 captions each. These are open-source public datasets. The datasets are originally developed for the English language therefore, they require preprocessing to make them usable for our research purpose.

3.1.2 Dataset Preprocessing

It is an integral part of this research because there are not any publicly available datasets for the Nepali language. Furthermore, the grammatical structure of the Nepali language is a lot more complex compared to the English language. In order to solve these problems and make our research more focused on the Nepali language, the following procedures are performed.

[1]`https://forms.illinois.edu/sec/1713398`

Caption conversion to Nepali The captions in the Flickr8k dataset are in the English language, therefore, to use these datasets in our context, they should be translated into the Nepali language. Such a translation is done using the Google translate API. Each line of the English caption file is translated and appended to a new text file.

Manual correction and annotation The translated texts using Google translate may contain various errors. The translations may not reflect the context of the image, thereby, generating incorrect captions or incurring a loss in the meaning of the descriptions of the image. We handle such problems through manual human corrections. The incorrect or garbage captions are either corrected or removed depending on their quality. It was not feasible to hire an expert therefore, we performed this task ourselves at the lab. Furthermore, the developed captions are randomly sampled to check for any irregularities.

Data Cleaning The translated and modified captions are further preprocessed to remove punctuations and numeric values. In this phase, all the unwanted characters and data from the captions are removed.

Generating Vocabulary and Text Vectorization A text vocabulary is generated from the translated captions by extracting all the unique words from the image description. In this work, a total of over 14,000 unique words are present in the vocabulary. Moreover, since the machines don't understand the natural language it must be converted to some numerical data to map each vocabulary word to a unique index value. This process is done using a built-in Text Vectorizer function available in the Keras library.

Dataset Creation The cleaned captions data are then split into three sets (Training set, Validation set, and Testing set) with 6000, 1000, and 1000 images respectively. The captions data are then mapped with the respective images and zipped together to create datasets for training and validation using the TensorFlow 'Dataset' library. The dataset created in this work can be found on Github[2].

3.2 Model Design and Implementation

The proposed system for Nepali image captioning comprises a CNN model for image feature extraction and a Transformer network for language modelling. Figure 1 shows the system architecture for this work. The CNN model can be developed from scratch but due to its advancement in recent years more accurate and efficient pre-trained models are available for use. Hence, 'EfficientNet' and 'ResNet101' are used to extract image features. These models are the pre-trained models trained on the Imagenet dataset. The CNN encodes the input image to a vector representation that is used by the decoder to generate captions. Since this work is not a classification task, the final softmax layer is removed from the CNN. Residual networks are deep neural networks that use the concept of skip connections to tackle the vanishing gradient problem (a problem where the gradient values of weights become very low for the machine to train efficiently) (He et al., 2015). ResNet-101 is a variant of a residual network with 101 layers, mainly composed of two blocks: Identity Block and Convolution Block. ResNet is one of the most used pre-trained CNN models in deep learning, especially in the Image Captioning domain. On the other hand, EfficientNet is a CNN model as well as a scaling technique that uses a set of preset scaling coefficients to uniformly scale depth, width, and resolution dimensions, in contrast to standard practice, which scales these variables arbitrarily (Tan and Le, 2019). There are 8 variants of EfficientNet (B0 - B7). EfficeintNetB0 is used for this research because of its simplicity and relatively good performance.

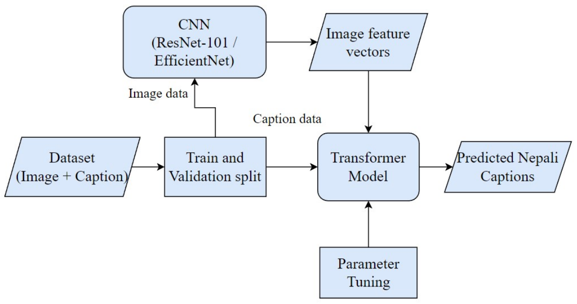

CNN
(ResNet-101 / EfficientNet)
Image feature vectors
Image data
Caption data
Dataset (Image + Caption)
Train and Validation split
Transformer Model
Predicted Nepali Captions
Parameter Tuning

Figure 1: Overall architecture of the system

The transformer model is used to generate captions instead of the conventional Recurrent Neural Network-based architecture because RNN doesn't support parallelization and transformer networks have outperformed RNN in NLP tasks in recent years. Our proposed system follows the transformer architecture proposed by (Vaswani et al.,

[2]`https://github.com/bipeshrajsubedi/Flickr8k_Nepali_Dataset`

Model	Architecture
Model A	ResNet-101 + Transformer
Model B	EfficientNetB0 + Transformer

Table 1: Different model architectures

2017). The main components of the transformer networks consist of the encoder, decoder, positional encoding, embeddings, softmax, and multi-headed attention. Attention is utilized in the transformer model to find the relevant collection of values based on a few keys and queries. Attention weights, which are derived using the encoder hidden state (Key) and decoder hidden state (Value), have recently been used to give priority to distinct encoder hidden states (values) in processing the decoder states (query) (Mishra et al., 2021). Single attention-weighted values have been found to be insufficient to capture the many features of the input. The transformer model thus employs multi-headed attention for tackling this challenge. Similarly, positional encoding is used by the transformer networks to keep information about the order of sequence by adding the relative or absolute position of the tokens in sequence (Vaswani et al., 2017). The positional encodings are added to the bottom of the encoder and decoder stacks.

In order to implement our model, the datasets generated are passed to both of the CNN-Transformer architectures mentioned in Table 1. The input images of size (299x299) are passed to the CNN encoder to generate image vectors. The image vectors are then passed to the transformer encoder. The transformer decoder part is fed with the respective captions to train the model. The encoder part comprises a single multi-headed attention head and a normalization layer whereas 2 multi-headed attention heads and 3 normalization layers are used in the decoder. These models are implemented using the TensorFlow Keras library. Table 2 shows the model parameters used in this work.

The model parameters are chosen based on explicit experimentation on our Nepali dataset. These are the optimal parameter values as per our research which can be further improved with the introduction of larger datasets and via parameter tuning.

4 Results and Discussions

The outcome of this work comprises the Nepali image captions dataset and experimental results of the model performance. A dataset of over

Parameters	Value
Image Size	(299,299)
Max. Vocab Size	15000
Sequence Length	20
Embedding Size	512
Batch Size	128
Optimizer	Adam
Loss Function	Categorical-crossentropy

Table 2: Model Parameters

40,000 Nepali image-caption pairs is created which are split into training, validation and testing sets. BLEU metric is used for quantitative analysis of the proposed system which is the most commonly used metric for text evaluation that shows the comparison of candidate translation with one or more reference translations (Google, 2022). Four BLEU scores (BLEU-1, BLEU-2, BLEU-3, and BLEU-4) are typically calculated in the context of image captioning. These scores evaluate merely matching grams of a certain order, such as single words (1-gram) or word pairs (2-gram or bigram), and so forth. BLEU score ranges from 0 to 1 where a score between 0.6 to 0.7 is considered to be the best achievable result but at the same time, a score between 0.3 to 0.4 is considered an understandably good translation and a score greater than 0.4 is considered high-quality translation(Google, 2022). In this work, BLEU score is calculated on the overall test data at once using the NLTK bleu library[3]. Table 3 shows the obtained results from this work as well as results from the existing works in Hindi and Bengali languages proposed by Mishra et al. (2021) and Shah et al. (2021) respectively. The obtained results demonstrate that Model B performs slightly better than Model A keeping the model parameters unchanged. The obtained scores imply that the proposed work has shown promising results and can be further improved in future. On the other hand, the first two BLEU scores are not as good compared to image captioning for Hindi and Bengali but can nevertheless serve as a reference for Nepali image captioning. It can also be seen that the last two BLEU scores are higher than that of Mishra et al. (2021) but lower than Shah et al. (2021). It is found that the predicted sentence generally describes the context of the image where its meaning is preserved but the words do not match with the

[3]https://www.nltk.org/api/nltk.translate.bleu_score.html

Model	B-1	B-2	B-3	B-4
Model A	0.49	0.40	0.37	0.34
Model B	0.52	0.42	0.37	0.34
Shah et al. (2021)	0.66	0.55	0.47	0.40
Mishra et al. (2021)	0.62	0.43	0.29	0.19

Table 3: Performance of different models

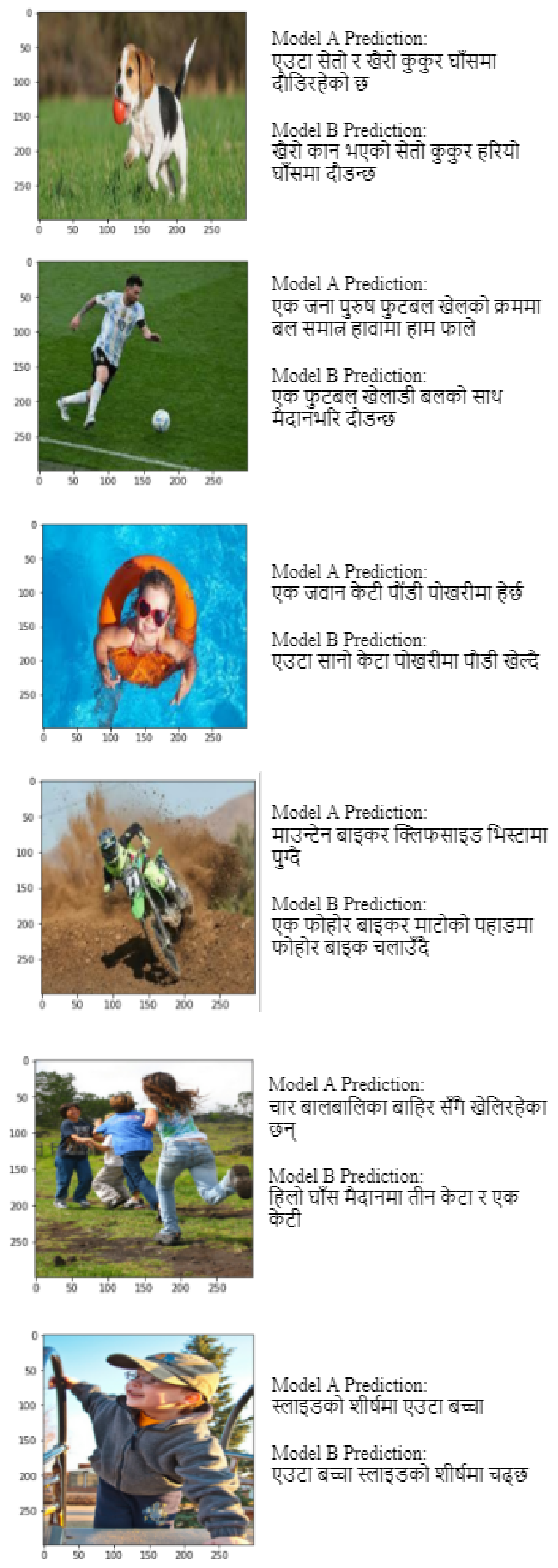
reference sentences in a specific order which leads to a lower BLEU score. Nevertheless, this is a pioneer work for Nepali image caption generation using transformers and hence can be used as a reference model. Some of the sample outputs of this work are shown in Figure 2.

5 Limitations

The proposed models have not acquired sufficient accuracy and are not able to generate the desired captions for all input images. Moreover, the datasets are also limited and only two pre-trained CNN models are considered for this work. Similarly, the hyperparameters may not be optimally tuned due to limited experimentations. Such limitations can affect the model's efficiency. We consider addressing them in future.

6 Conclusion and Future works

In this research work, a CNN-Transformer-based Nepali Image Captioning system is implemented. At first, the Flickr8k datasets are translated and preprocessed to create a Nepali captions dataset. The datasets are then fed to two models: Model A and Model B with the same model parameters. The outcome of this experiment shows promising results. Model B performed slightly better than Model A on 40,455 Nepali image-caption pairs. Moreover, these models are able to generate captions from the given input image. On the other hand, this work has some limitations as well which can be addressed in the future. The experimentation on larger datasets and fine-tuning of the hyperparameters can be performed in future which are expected lead to better results. MS COCO and Flickr30k datasets can be used after preprocessing for this purpose. Similarly, other CNN models such as InceptionV3, Xception, EfficientNetB7 etc. can be explored for image captioning tasks. Furthermore, this research work can be extended to video captioning for the Nepali language as well.



Model A Prediction:
एउटा सेतो र खैरो कुकुर घाँसमा दौडिरहेको छ

Model B Prediction:
खैरो कान भएको सेतो कुकुर हरियो घाँसमा दौडन्छ

Model A Prediction:
एक जना पुरुष फुटबल खेलको क्रममा बल समात्न हावामा हाम फाले

Model B Prediction:
एक फुटबल खेलाडी बलको साथ मैदानभरि दौडन्छ

Model A Prediction:
एक जवान केटी पौंडी पोखरीमा हेर्छ

Model B Prediction:
एउटा सानो केटा पोखरीमा पौडी खेल्दै

Model A Prediction:
माउन्टेन बाइकर क्लिफसाइड भिस्टामा पुग्दै

Model B Prediction:
एक फोहोर बाइकर माटोको पहाडमा फोहोर बाइक चलाउँदै

Model A Prediction:
चार बालबालिका बाहिर सँगै खेलिरहेका छन्

Model B Prediction:
हिलो घाँस मैदानमा तीन केटा र एक केटी

Model A Prediction:
स्लाइडको शीर्षमा एउटा बच्चा

Model B Prediction:
एउटा बच्चा स्लाइडको शीर्षमा चढ्छ

Figure 2: Sample results obtained from our proposed models

References

Aashish Adhikari and Sushil Ghimire. 2019. Nepali image captioning. In *2019 Artificial Intelligence for Transforming Business and Society (AITB)*, volume 1, pages 1–6.

Amit Saha Ami, Mayeesha Humaira, Md Abidur Rahman Khan Jim, Shimul Paul, and Faisal Muhammad Shah. 2020. Bengali image captioning with visual attention. In *2020 23rd International Conference on Computer and Information Technology (ICCIT)*, pages 1–5.

Google. 2022. Evaluating models.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Santosh Kumar Mishra, Rijul Dhir, Sriparna Saha, Pushpak Bhattacharyya, and Amit Kumar Singh. 2021. Image captioning in hindi language using transformer networks. *Computers Electrical Engineering*, 92:107114.

Md Aminul Haque Palash, M. D. Abdullah Al Nasim, Sourav Saha, Faria Afrin, Raisa Mallik, and Sathishkumar Samiappan. 2021. Bangla image caption generation through cnn-transformer based encoder-decoder network. *CoRR*, abs/2110.12442.

Faisal Muhammad Shah, Mayeesha Humaira, Md Abidur Rahman Khan Jim, Amit Saha Ami, and Shimul Paul. 2021. Bornon: Bengali image captioning with transformer-based deep learning approach. *SN Computer Science*, 3.

Xiangqing Shen, Bing Liu, Zhou Yong, and Jiaqi Zhao. 2020. Remote sensing image caption generation via transformer and reinforcement learning. *Multimedia Tools and Applications*, 79.

Lakshminarasimhan Srinivasan and Dinesh Sreekanthan. 2018. Image captioning-a deep learning approach.

Mingxing Tan and Quoc V. Le. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946.

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.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044.

Verb Phrase Anaphora: *Do(ing) so* with Heuristics

Sandhya Singh[*], **Kushagra Shree**[†], **Sriparna Saha**[†],
Pushpak Bhattacharyya[*], **Gladvin Chinnadurai**[‡], **Manish Kumar Vatsa**[‡]
[*]CFILT, Indian Institute of Technology Bombay, [†]Indian Institute of Technology Patna,
[‡]LG Soft India
{sandhyasingh, pb}@cse.iitb.ac.in,
kushagra4xps@gmail.com, sriparna@iitp.ac.in,
{gladvin.durai, manish.vatsa}@lge.com

Abstract

Verb Phrase Anaphora (VPA) is a universal language phenomenon. It can occur in the form of *do so* phrase, *verb phrase ellipsis, etc.* Resolving VPA can improve the performance of Dialogue processing systems, Natural Language Generation (NLG), Question Answering (QA) and so on. In this paper, we present a novel computational approach to resolve the specific verb phrase anaphora appearing as *do so* construct and its lexical variations for the English language. The approach follows a heuristic technique using a combination of parsing from classical NLP, state-of-the-art (SOTA) Generative Pre-trained Transformer (GPT) language model and RoBERTa grammar correction model. The result indicates that our approach can resolve these specific verb phrase anaphora cases with 73.40 F1 score. The data set used for testing the specific verb phrase anaphora cases of *do so* and *doing so* is released for research purposes. This module has been used as the last module in a coreference resolution pipeline for a downstream QA task for the electronic home appliances sector.

1 Introduction

Anaphora is a linguistic construct used for maintaining coherence in the text without being repetitive. A solution to Anaphora Resolution (AR) can improve the performance of downstream tasks like Dialogue systems, Natural Language Generation (NLG), Question Answering *etc.* The process of identifying the linguist element *(anaphor)* that is referring to a preceding linguistic element *(antecedent)* in the context is known as *Anaphora Resolution (AR).*

According to Mitkov (2002), anaphoras can be classified as pronominal anaphora, lexical noun phrase anaphora, verb phrase anaphora, adverb anaphora and zero anaphora. The current state-of-the-art systems (Clark and Manning, 2016; Lee et al., 2017; Joshi et al., 2019a) have obtained high accuracy for the most prevalent type of anaphoras *i.e.* pronominal anaphora and lexical noun phrase anaphora cases. However, verb phrase anaphora, adverb anaphora and zero anaphora still remain unsolved due to the complexities involved in these language phenomena. This paper deals with one such case: **verb phrase anaphora (VPA).** The verb phrase anaphor is resolved by a preceding verb phrase plus any complement and adjunct as the antecedent. The verb phrase anaphor constructs occur as a combination of *so, this, that, it and the same thing* along with *do*. *Example 1* shows the verb phrase anaphor *doing so* referring to the verb phrase *use energy saver mode* as an antecedent. Besides antecedent identification, the grammar also enforces a syntactic modification to the antecedent as *"Using energy saver mode"* for resolution in reference to the anaphor.

> ***[Use energy saver mode]$_1$*** *in the air conditioner.*
> ***[Doing so]$_1$*** *helps reduce the load on the pocket.*
>
> *Antecedent substituted output:* **Using energy saver mode** helps reduce the load on the pocket.
> **Example 1**

In this paper, we focus on the specific case of *so anaphora* which is used in conjunction with the verb form *do*. This construct is one of the most frequent forms of verb phrase anaphora. The challenges posed by *do so* constructs are both semantic and morphosyntactic in nature. **The contributions** through this work are listed here as:

- A novel computational heuristic approach using a combination of classical NLP, transformer-based language model and a grammar correction model to resolve specific *do so* constructs as the anaphoric expression.
- A dataset of 350 data points of *do so* VPA constructs in inter-sentence and intra-sentence format is also released as a part of our research contribution.

The paper is organized as follows: Section 2 discusses the syntactic and semantic challenges

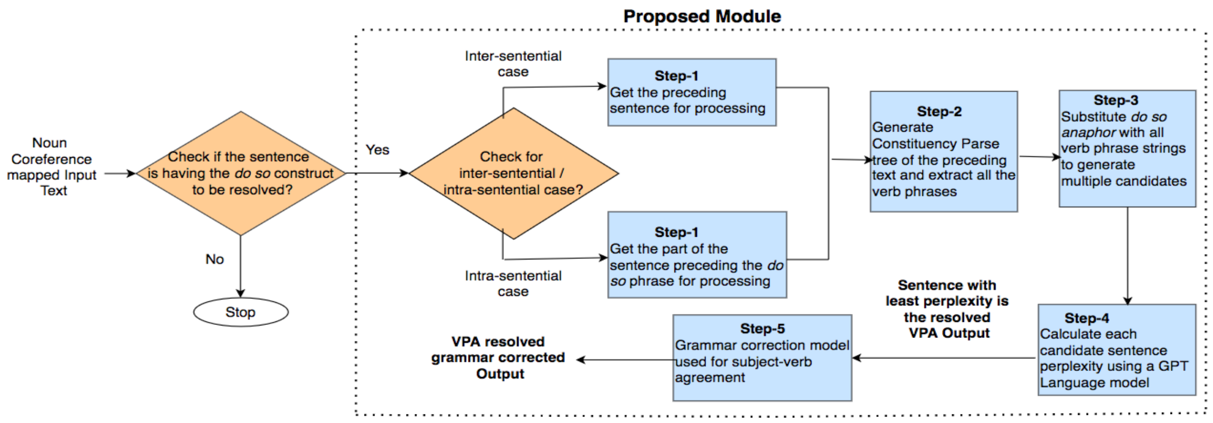

Proposed Module
Noun Coreference mapped Input Text
Check if the sentence is having the do so construct to be resolved?
Yes
No
Stop
Check for inter-sentential / intra-sentential case?
Inter-sentential case
Intra-sentential case
Step-1 Get the preceding sentence for processing
Step-1 Get the part of the sentence preceding the do so phrase for processing
Step-2 Generate Constituency Parse tree of the preceding text and extract all the verb phrases
Step-3 Substitute do so anaphor with all verb phrase strings to generate multiple candidates
Step-4 Calculate each candidate sentence perplexity using a GPT Language model
Sentence with least perplexity is the resolved VPA Output
Step-5 Grammar correction model used for subject-verb agreement
VPA resolved grammar corrected Output

Figure 1: Proposed approach flow as the last module of a coreference resolution pipeline for a QA system.

associated with *do so* anaphor. Section 3 presents the related literature with specific focus on verb phrase anaphora. Section 4 describes our dataset and the heuristic approach used to resolve verb phrase anaphora. Section 5 analyses the results obtained followed by conclusion and future work.

2 Challenges with *do so* construct

Anaphora and ellipsis are preferred linguistic mechanisms used in conversation. In general, any anaphora resolution is a three step process: *i) Anaphor identification, ii) Antecedent candidates identification, and iii) Choosing the most likely antecedent candidate.* The general resolution technique involves eliminative constraints based on gender, number, semantic consistency and weighting preference based on proximity, centering, syntactic/semantic (role) parallelism between the anaphor and antecedent (Sayed, 2003). But the verb phrase anaphora is found to be resolved in a more complex manner.

The anaphoric verb phrases such as *do it, do that, do so, ellipsis etc.* are known to inherit the properties from their referent. As in *example 2*, the phrase *did too* not only refers to the event of eating performed by John but also to properties such as *quietly* and *from the plate*.

*John **[quietly ate the cake from the plate]**$_1$. Jerry **[did too.]**$_1$*
Antecedent substituted output: Jerry **quietly ate the cake from the plate** too.
Example 2

Verb phrase anaphora when resolved from a discourse perspective departs on two aspects from the standard approaches of entity resolution (Prüst et al., 1994). First, it cannot be determined by simply identifying the anaphoric verb phrase with an antecedent verb phrase. The resolution process must establish a syntactic/semantic parallelism between clauses or discourse constituent units in which the verb phrase occurs. Second, the discourse structure significantly influences the reference possibilities of verb phrase anaphora.

As we are working on a QA system for the home appliances domain, we are faced with VPA constructs in the user manual for the devices. Thus, our work focuses specifically on *do so* anaphora construct in the QA problem which we are trying to solve as a goal. It has been identified that *do so* does not refer to only the verb alone but the entire verb phrase consisting of the main verb, auxiliary verb, its complements and adjuncts as constituted in phrase structure grammar. For anaphora resolution, both terms share the load with *do* placing the semantic requirement on the antecedent whereas *so* is responsible for the anaphoric work.

*My grandmother **[knows all her grandchildren's names]**$_1$, and she manages to **[do so]**$_1$ despite her Alzheimer's.*
Example 3

*The students, who **[know French best]**$_1$, **[do so]**$_1$ because they lived in France for a year.*
Example 4

The *do so* construct also appears as the infinitive form as *"to know..."* in *example 3* and in the form where the antecedent is contained in a relative clause *"who know French ..."*, thus modifying the subject of *do so* as in *example 4*. This syntactic and semantic analysis of the construct highlights the challenges it poses in resolving it.

3 Related Work

The research on computational anaphora and coreference resolution has seen a paradigm shift from heuristic approaches to machine learning ap-

Data Points	Inter-sentential	Intra-sentential
350	210 (60%)	140 (40%)

(a) Distribution of inter-sentential & intra-sentential cases

Construct Type	Data Points
Doing so construct	130
Do so construct	149
Others (*did, does, these, this*)	71

(b) Distribution of different types of verb anaphor cases

Table 1: Data distribution statistics of our dataset

proaches in both nominal-antecedent anaphora (Ng, 2010) and non nominal-antecedent anaphora (Kolhatkar et al., 2018) categories.

Specific to VPA, considerable work is seen in the field of theoretical linguistics for different languages. Hankamer and Sag (1976) investigated verb phrase anaphora as deep or surface anaphora for the English language and Houser et al. (2006) studied the same for the Danish language. Later, Houser (2010) and Wei and hui Audrey Li (2016) studied the syntactic and semantic challenges of *do so* construct for the English and Mandarin language respectively. In dialogue systems, the problem of ellipsis has been addressed by the use of a supervised discriminative machine learning model (Kenyon-Dean et al., 2016) and joint modelling with coreference and question-answering data (Aralikatte et al., 2021). Liu et al. (2016) explored the decomposition of verb phrase ellipsis resolution into computational subtasks. Itegulov and Lebedeva (2018) experimented with identifying dependent type events for verb phrase anaphora resolution. Marasović et al. (2017) used an LSTM-Siamese Net mention-ranking model to learn abstract anaphora resolution or discourse deixis.

On the data front, datasets like OntoNotes (Pradhan et al., 2012), WikiCoref (Ghaddar and Langlais, 2016) *etc.* are the common datasets used for benchmarking the nominal-antecedent anaphora and coreference resolution models. Other datasets specifically addressing the verb phrase anaphora cases are AARAU Corpus (Poesio et al., 2018), CODI-CRAC 2021 Shared Task corpus (Khosla et al., 2021) and VP ellipsis corpus (Bos and Spenader, 2011). Though few instances of *do so* constructs are available in VPE corpus (Bos and Spenader, 2011), we could not use it for our experiment as our downstream QA system deals with a specific pattern of *do so* construct.

As the VPA constructs are consistently coming to the forefront of Dialogue systems and QA systems as a challenge, it motivated us to explore the specific case of *do so* VPA construct and create a dataset to manage our specific needs.

4 Experiment

4.1 Dataset

During the process of solving the language generation problem of a QA system, it has been noticed that the SOTA entity resolution system (Joshi et al., 2019b) is not able to resolve VPA cases. Since the SOTA model is trained on the OntoNotes (Pradhan et al., 2012) dataset, we explored its dataset and guidelines. The coreference guidelines of OntoNotes clearly state that verb is to be marked as a single-word span only if it is coreferenced with an existing noun phrase. With this guideline, it can be inferred that the deep learning model trained on this dataset will not be able to resolve the *do so* type of verb anaphoras as the dataset is not annotated to address these cases of VPA.

For our task, we constructed a targeted dataset of 350 data points with surface variations of *do so* VPA constructs, *viz. doing so, does so, did so*. The dataset contains both kinds of cases where the scope of the antecedent is either inter-sentential or intra-sentential. The data points of *do so* VPA constructs are collated from two sources: our QA system which we are automating and BNC corpus[1]. An equal number of data points were collated to balance the data for generic VPA resolution and coverage.

Two annotators helped us annotate the antecedent span for each VPA in a standoff annotation format. The antecedent span is annotated as clusters with word index based on subword tokenizer (Joshi et al., 2019b) output and stored in dictionary format.

A kappa score of 0.89 indicates a high Inter Annotator Agreement (IAA) for the antecedent spans. The detailed data stats are given in table 1. The evaluation and analysis in this paper are done on our dataset. The dataset is released as part of the contribution to further VPA research[2].

[1]BNC Consortium, The British National Corpus, XML Edition, 2007, Oxford Text Archive, http://hdl.handle.net/20.500.12024/2554.

[2]`https://github.com/Sandhya2207/VPA_dataset`

Data	MUC			B-Cube			CEAF-e			F1-avg
	R	P	F1	R	P	F1	R	P	F1	
Baseline	37.43	77.38	50.46	35.24	77.9	48.53	32.5	79.97	46.22	48.40
Our approach	68.94	66.07	**67.47**	74.67	71.91	**73.27**	80.79	78.2	**79.47**	**73.4**

Table 2: Evaluation Result of output from our approach. (R: recall, P: precision, F1: F1 score)

Input Stage	**Process**	**Input**	**Output**
SOTA model Output	Coreference Resolved using SOTA	Shawn turned on the killer machine to kill mosquitoes. Doing so, he says, narrows its prospects for survival.	Shawn turned on the killer machine to kill mosquitoes. Doing so, Shawn says, narrows mosquitoes' prospects for survival.
Step-1	Identify the span of probable VPA	Shawn turned on the killer machine to kill mosquitoes. Doing so, Shawn says, narrows mosquitoes' prospects for survival.	S1: Shawn turned on the killer machine to kill mosquitoes. S2: Doing so, Shawn says, narrows mosquitoes' prospects for survival.
Step-2	Get the constituency parse string and consider only verb phrase	S1: Shawn turned on the killer machine to kill mosquitoes.	["turned on the killer machine to kill mosquitoes", " to kill mosquitoes", " kill mosquitoes"]
Step-3	Substitute verb anaphor with verb phrases to get candidate antecedents	["turned on the killer machine to kill mosquitoes", " to kill mosquitoes", " kill mosquitoes"]	["turned on the killer machine to kill mosquitoes, Shawn says, narrows mosquitoes prospects for survival. ", "to kill mosquitoes, Shawn says, narrows mosquitoes' prospects for survival. ", "kill mosquitoes, Shawn says, narrows mosquitoes' prospects for survival."]
Step-4	Get probability of correct sentence using GPT LM and pick the sentence with lowest perplexity	["turned on the killer machine to kill mosquitoes, Shawn says, narrows mosquitoes' prospects for survival. ", "to kill mosquitoes, Shawn says, narrows mosquitoes prospects for survival. ", "kill mosquitoes, Shawn says, narrows mosquitoes' prospects for survival."]	shawn turned on the killer machine to kill mosquitoes . turned on the killer machine to kill mosquitoes, shawn says, narrows mosquitoes' prospects for survival .
Step-5	Correct the grammar for subject-verb agreement	shawn turned on the killer machine to kill mosquitoes . turned on the killer machine to kill mosquitoes, shawn says, narrows mosquitoes' prospects for survival.	shawn turned on the killer machine to kill mosquitoes . turning on the killer machine to kill mosquitoes, shawn says, narrows mosquitoes' prospects for survival .

Table 3: Input and Output of each step of our approach as shown in figure 1

4.2 Our Approach

The coreference module of our QA system works in stages. It begins with resolving all nominal antecedent cases using the SOTA coreference model (Joshi et al., 2019b). The nominal coreference clusters identified by the SOTA model are mapped to the input sentence to get noun coreference mapped output text. This text forms the input to our VPA module.

Figure 1 shows the flow of our proposed pipeline approach. In *step 1*, the nominal coreference mapped input text is checked for the presence of *do so* and all its lexical forms, *viz. doing so, does so, did so.* If the lexical text matches *do so* construct, it is a candidate for VPA resolution and is further checked for intra/inter sentential case based on the location of *do so* construct. For inter-sentential cases, the sentence preceding the sentence having the *do so* phrase is considered as the scope of its VPA antecedent. And, for the intra-sentential case, part of the sentence preceding the *do so* phrase is considered as the scope of its VPA antecedent.

In *step 2*, a constituency parse tree of the VPA antecedent text from *step 1* is generated using both Stanford CoreNLP parser (Klein et al., 2003; Manning et al., 2014) and constituency parser with ELMo embeddings (Joshi et al., 2018) for improved coverage.

From the generated parse tree, all the verb phrases are extracted as possible antecedent candidates and mapped in place of *do so* phrase to get anaphora resolved candidates as in *step 3*. This step leads to multiple candidates for identifying the correct antecedent for VPA.

At *step 4*, we get the probability of each candidate sentence using the generative pre-trained transformer (GPT) language model (Radford et al., 2018). The intuition here is that a syntactically correct candidate sentence will have higher probability as compared to incorrect candidate sentence. Using sentence probability we calculate the perplexity of all the candidate sentences. The candidate sentence with the lowest perplexity is considered as the antecedent resolved VPA output sentence.

Since subject-verb agreement is required for the correctness of the sentence in VPA, a pre-trained RoBERTa grammar correction model (Omelianchuk et al., 2020) is used to get subject-verb agreement in antecedent mapped text in *step 5*. Table 3 shows the module output after each step. Since the output of our system is to be consumed by machines, the naturality of the sentence was less of a concern.

S. No.	Input	Baseline Output	Our Approach Output	Reference Output	Remarks
1	Never put your money in a sinking company . In plainer terms , failure to do so leads to loss .	Never put your money in a sinking company . In plainer terms , failure to do so leads to loss .	Never put your money in a sinking company. In plainer terms , failure to put your money in a sinking company leads to loss .	Never put your money in a sinking company. In plainer terms, putting your money in a sinking company leads to loss .	Meaning changed for negative sentence.
2	A dolphin that watches a model place a ball in a basket might place the ball in the basket when asked to mimic t he behavior, but it may do so in a different manner.	A dolphin that watches a model place a ball in a basket might place a ball in a basket when asked to mimic the behavior, but A dolphin that watches a model place a ball in a basket may do so in a different manner.	A dolphin that watches a model place a ball in a basket might place a ball in a basket when asked to mimic the behavior, but A dolphin that watches a model place a ball in a basket may place a ball in a basket when asked to mimic the behavior , but A dolphin that watches a model place a ball in a basket may in a different manner.	A dolphin that watches a model place a ball in a basket might place the ball in the basket when asked to mimic the behavior, but dolphin may mimic the behavior in a different manner.	In case of multiple verb phrases as antecedent, not able to pick the accurate verb phrase. But the Verb phrase is available in top 3 choices.

Table 4: Error analysis of the output from our approach

5 Result

The standard evaluation metrics used for anaphora resolution is link based MUC score (Vilain et al., 1995), mention based B^3 score (Bagga and Baldwin, 1998) and optimal mapping based CEAF-E score (Luo, 2005). We evaluated our result using the standard CoNLL 2012 metric (Pradhan et al., 2012) which is calculated as an average of MUC, B-cube and CEAF metrics. Table 2 shows the precision, recall and F1 score for each metric and their average score. The baseline used for comparison is the output from the state-of-the-art BERT e2e-coreference model (Joshi et al., 2019b).

The high recall value for MUC, B-cube and CEAF metrics indicates that our approach is able to identify the antecedent and its span with higher accuracy over the baseline model. And, the overall average F1 score is showing an improvement of 25.0 value over the baseline. Table 3 shows the input and output of each step in our module.

5.1 Qualitative Analysis

Table 4 shows the error analysis of some output cases. It shows that our approach is not able to manage the following sentence formats.

- Negative sentences are not semantically correct after VPA mapping as in row 1 of the table.
- In case of multiple verb phrases in antecedents, our approach is not able to identify the boundary of prospective antecedent as in row 2 of the table.

The innovative sentence constructs at the intra-sentence level, the cataphor constructs and relative clause constructs are still an open problem to be addressed.

6 Conclusion and Future work

This paper presents a computational heuristic approach to resolve the *do so* verb phrase anaphora. The approach uses a constituency parser to get all the syntactic components of the text. From the syntactic components, all the verb phrases from preceding text is substituted in place of the verb anaphor to generate the candidate sentences. A pre-trained language model is used to select the most probable antecedent.

The result shows that our approach can identify the antecedent and its span with good accuracy on the VPA dataset developed for this experiment. The dataset used will be shared for further research. In the future, we plan to resolve the span identification issue in the intra-sentential case of *do so* construct where no conjunct is used as found in our error analysis. we also plan to investigate if our approach can be extended to other verb phrase anaphora constructs in the English language.

References

Rahul Aralikatte, Matthew Lamm, Daniel Hardt, and Anders Søgaard. 2021. Ellipsis resolution as question answering: An evaluation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 810–817.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *The first international conference on language resources and evaluation workshop on linguistics coreference*, volume 1, pages 563–566. Citeseer.

Johan Bos and Jennifer Spenader. 2011. An annotated corpus for the analysis of vp ellipsis. *Language resources and evaluation*, 45(4):463–494.

Kevin Clark and Christopher D. Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. In *Empirical Methods on Natural Language Processing*.

Abbas Ghaddar and Philippe Langlais. 2016. Wikicoref: An english coreference-annotated corpus of wikipedia articles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 136–142.

Jorge Hankamer and Ivan Sag. 1976. Deep and surface anaphora. *Linguistic inquiry*, 7(3):391–428.

Michael Houser. 2010. The syntax and semantics of do so anaphora.

Michael J Houser, Line Mikkelsen, Maziar Toosarvandani, Erin Brainbridge, and Brian Agbayani. 2006. Verb phrase pronominalization in danish: Deep or surface anaphora? *To appear in the Proceedings of WECOL*, 2007.

Daniyar Itegulov and Ekaterina Lebedeva. 2018. Handling verb phrase anaphora with dependent types and events. In *International Workshop on Logic, Language, Information, and Computation*, pages 210–222. Springer.

Mandar Joshi, Omer Levy, Daniel S. Weld, and Luke Zettlemoyer. 2019a. BERT for coreference resolution: Baselines and analysis. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Mandar Joshi, Omer Levy, Daniel S Weld, and Luke Zettlemoyer. 2019b. Bert for coreference resolution: Baselines and analysis. *arXiv preprint arXiv:1908.09091*.

Vidur Joshi, Matthew Peters, and Mark Hopkins. 2018. Extending a parser to distant domains using a few dozen partially annotated examples. In *Proceedings of the Association for Computational Linguistics*, pages 1190–1199, Melbourne, Australia. Association for Computational Linguistics.

Kian Kenyon-Dean, Jackie Chi Kit Cheung, and Doina Precup. 2016. Verb phrase ellipsis resolution using discriminative and margin-infused algorithms. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1734–1743.

Sopan Khosla, Juntao Yu, Ramesh Manuvinakurike, Vincent Ng, Massimo Poesio, Michael Strube, and Carolyn Rosé. 2021. The codi-crac 2021 shared task on anaphora, bridging, and discourse deixis in dialogue. In *Proceedings of the CODI-CRAC 2021 Shared Task on Anaphora, Bridging, and Discourse Deixis in Dialogue*, pages 1–15.

Dan Klein, Christopher D Manning, et al. 2003. Fast exact inference with a factored model for natural language parsing. *Advances in neural information processing systems*, pages 3–10.

Varada Kolhatkar, Adam Roussel, Stefanie Dipper, and Heike Zinsmeister. 2018. Anaphora with non-nominal antecedents in computational linguistics: A survey. *Computational Linguistics*, 44(3):547–612.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. *EMNLP*.

Zhengzhong Liu, Edgar Gonzalez, and Dan Gillick. 2016. Exploring the steps of verb phrase ellipsis.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 25–32.

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.

Ana Marasović, Leo Born, Juri Opitz, and Anette Frank. 2017. A mention-ranking model for abstract anaphora resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 221–232, Copenhagen, Denmark. Association for Computational Linguistics.

Ruslan Mitkov. 2002. *Anaphora resolution*. Pearson Education.

Vincent Ng. 2010. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 1396–1411.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR – grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA â†' Online. Association for Computational Linguistics.

Massimo Poesio, Yulia Grishina, Varada Kolhatkar, Nafise Sadat Moosavi, Ina Roesiger, Adam Roussel, Fabian Simonjetz, Alexandra Uma, Olga Uryupina, Juntao Yu, et al. 2018. Anaphora resolution with the arrau corpus. In *Proceedings of the First Workshop on Computational Models of Reference, Anaphora and Coreference*, pages 11–22.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 1–40.

Hub Prüst, Remko Scha, and Martin Van Den Berg. 1994. Discourse grammar and verb phrase anaphora. *Linguistics and Philosophy*, 17(3):261–327.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Imran Q Sayed. 2003. Issues in anaphora resolution. *Stanford*.

Marc Vilain, John D Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Sixth Message Understanding Conference (MUC-6): Proceedings of a Conference Held in Columbia, Maryland, November 6-8, 1995*.

Ting-Chi Wei and Yen hui Audrey Li. 2016. How to do so in mandarin chinese. *Journal of East Asian Linguistics*, 25(2):183–212.

Event Oriented Abstractive Summarization

Aafiya Hussain
K.J. Somaiya College of Engineering
aafiya.h@somaiya.edu

Talha Chafekar
K.J. Somaiya College of Engineering
talha.c@somaiya.edu

Grishma Sharma
K.J. Somaiya College of Engineering
grishma.sharma@somaiya.edu

Deepak Sharma
K.J. Somaiya College of Engineering
deepaksharma@somaiya.edu

Abstract

Abstractive Summarization models are generally conditioned on the source article. This would generate a summary with the central theme of the article. However, it would not be possible to generate a summary focusing on specific key areas of the article. To solve this problem, we introduce a novel method for abstractive summarization. We aim to use a transformer to generate summaries which are more tailored to the events in the text by using event information. We extract events from text, perform generalized pooling to get a representation for these events and add an event attention block in the decoder to aid the transformer model in summarization. We carried out experiments on CNN / Daily Mail dataset and the BBC Extreme Summarization dataset. We achieve comparable results on both these datasets, with less training and better inclusion of event information in the summaries as shown by human evaluation scores.

1 Introduction

Summarization is the process of giving an overview of a piece of text. This is done to reduce the amount of time required to understand a topic by eliminating information that is not as relevant to the topic. In abstractive summarization, the model tries to grasp the source text and produce a summary that consists of novel words and phrases. As the sentences produced are generated by the model, the redundancy in the final summary is significantly reduced as compared to extractive text summarization. This task of summarization is a complex one for humans as well. The difficulty of this task is due to the fact that summarization is fairly subjective. People may assign importance to parts of text differently. Thus, the main focus of one person's summary may be just a passing mention in someone else's summary. Another reason for this difficulty is that there has to be a balance between novel text and text taken from the source article. For abstractive summaries we want the model to understand the source text, and then represent it in a concise manner. This is a tricky balance to maintain as we want to achieve saliency, but we also want to avoid direct copying from the source text. We use transformers to carry out abstractive text summarization on the CNN / Daily Mail dataset (Nallapati et al., 2016) and Extreme Summarization dataset (Narayan et al., 2018). The transformer architecture we have used is BART (Lewis et al., 2020).

In this paper, we propose a system, which modifies the existing BART (Lewis et al., 2020) architecture by adding an additional event attention block. Events can be described as the sub-topics around which the news articles revolve. Identifying these events and adding them separately, along with source text, prompts the model to focus the summaries around these events. We perform keyphrase extraction using KeyBERT (Grootendorst, 2020) to extract important events from the source text and use these events for prompting our model to generate event-oriented summaries.

We achieve comparable results for ROUGE (Lin, 2004) and BERTScore (Zhang* et al., 2020) metrics for CNN / Daily Mail and XSum datasets, with the base variant of the BART model. Moreover, with the help of human evaluation, we quantify the extent to which our generated summaries are influenced by the events input to the model.

2 Background

2.1 Problem Statement

Given an input document $\mathbf{X} = \mathbf{x_1}, ..., \mathbf{x_n}$, we aim to generate a summary $\mathbf{Y}' = \mathbf{y}'_\mathbf{1}, ..., \mathbf{y}'_\mathbf{m}$ where $\mathbf{n}$ and $\mathbf{m}$ denote article and summary lengths respectively. The summary is generated in reference to $\mathbf{Y} = \mathbf{y_1}, ..., \mathbf{y_p}$ where $\mathbf{p}$ is the length of the ground truth summary. We make use of auxiliary input, i.e. event tokens $\mathbf{E} = \mathbf{e_1}, ..., \mathbf{e_b}$ consisting of $\mathbf{b}$ events,

Ground Truth: The rapper assaulted the photographer at Los Angeles International Airport in 2013. West apologized as part of the settlement, the photographer's lawyer says. **Generated summary:** Kanye West has settled a lawsuit with a paparazzi photographer he assaulted. Daniel Ramos had filed the civil suit against West after the hip-hop star attacked him and tried to wrestle his camera from him. **Events:** west has settled lawsuit, civil suit against west, ramos had filed the, photographer he assaulted
Ground Truth: New research finds direct link between exam stress and performance. London headteacher Michael Ribton says revision plans, flashcards and cram techniques can all help children prepare for the exam season. He advises parents that extra tuition and bribes shouldn't be necessary. **Generated summary:** Study by Lancashire's Edge Hill University and the University of South Australia found a direct link between anxiety and performance. Pupils who worry about their exam performance are more likely to do badly than those who are less anxious. **Events:** exam performance are more, worry about their exam, between anxiety and performance, exams and grades achieved
Ground Truth: Smoke from massive fires in Siberia created fiery sunsets in the Pacific Northwest. Atmospheric winds carried smoke from the wildfires across the Pacific Ocean. Smoke particles altered wavelengths from the sun, creating a more intense color. **Generated summary:** A fiery sunset greeted people in Washington Sunday. The deep reddish color caught Seattle native Tim Durkan's eye. **Events:** fiery sunset greeted people, siberia the dramatic sunset, reddish color caught seattle, sunset began showing up
Ground Truth: Villagers in Shangdong are seen using bags as long as six metres. It is becoming a common behaviour in some villages since 2011. Previous investigation suggested gas in the bag are often stolen. Gas carriers have little understanding of dangers claiming it to be safe. **Generated summary:** Residents from Lijin village in Dongying city carry the explosive in bags as long as six metres on rickshaws. Worried passers-by compared this behaviour to carrying a bomb on their backs. **Events:** the explosive in bags, stealing gas from large, gas this reckless behaviour, carrying bomb on their

Table 1: Few results for samples from CNN DailyMail dataset.

with each event having a fixed number of tokens **k**. We make use of seq2seq architecture, specifically a transformer encoder and decoder network, along with input prompting, generalized pooling and an additional event attention block which focuses on event embeddings to generate event-oriented summaries.

2.2 Sequence Models

Since abstractive summarization is a sequence based task, the initial application of deep learning models to abstractive summarization started with an attention based encoder and a sequence decoder making use of beam search (Rush et al., 2015). Since the decoder was not a recurrent model, later approaches to recurrent based summarization systems (Nallapati et al., 2016) performed better in generating summaries. The encoder mechanism consisted of an attention block attending to different encoder time steps, and a decoder RNN, which would take into account the encoder's attention outputs for the decoding step. A pointer generator network model (See et al., 2017) was introduced, which would dynamically decide whether to generate new tokens or to copy tokens from the article text, thus making the summarization model more factually correct. However, with the rise of transfer learning, language models like BERT (Devlin et al., 2019) and GPT (Radford et al., 2018) gave a far better performance as compared to LSTMs for the same task.

2.3 Transformer Models

With the introduction of transformers (Vaswani et al., 2017), sequence to sequence tasks have become much easier using pre-training. Transformers were first used for training on machine translation tasks on the WMT 2014 English-French dataset (Bojar et al., 2014). Transformers outperformed previous models on the BLEU metric (Papineni et al., 2002). Consequently, the application of transformers to summarization was done by a BERT encoder used to feed embeddings to a transformer decoder (Zhang et al., 2019). A two stage mechanism, where masked language modelling as used in BERT (Devlin et al., 2019) is applied for refined word prediction in the later stage of the model. Raffel et al. (2020) pre-trained a transformer model on the C4 dataset, along with an analysis of different pre-training objectives such as prefix language modelling. PEGASUS (Zhang et al., 2020) follows the same pre-training objective as BERT, however, they introduce a summarization specific objective, i.e. to mask and generate sentences, similar to an extractive summary. Results indicated a significant increase in ROUGE scores with previous SOTA methods. Another training objective proposed was denoising in BART (Lewis et al., 2020), i.e. corrupting the input sequences with a range of operations including replacement, masking, text infilling, and sentence permutation.

3 Related Work

3.1 Event Extraction

Örs et al. (2020) use pre-trained transformer models, namely BERT (Devlin et al., 2019) and ALBERT (Lan et al., 2020) for predicting if a pair of sentences point to the same event, and later use the prediction scores to capture the degree of relatedness between different sentence pairs. Xu et al. (2021) propose a graph-based model to capture the relation between different sentences and entity mentions. A tracker module is used which stores the global information about the extracted events, which can be used to query the stored information for interdependency relations. Rule-based systems (Ritter et al., 2012; Valenzuela-Escárcega et al., 2015) follow a syntactic and a word feature-based approach for extraction of events. A more general approach is used by Sun et al. (2021) where a multi-task training objective is followed over pre-trained language model embeddings for n-grams to capture both their informativeness and phraseness. Instead of following a complex method, we use KeyBERT (Grootendorst, 2020), which is more simple and minimalistic as it uses BERT embeddings and cosine similarity. Moreover, we are able to set the hyperparameters for keyphrase extraction such as keyphrase n-gram length, and the number of keyphrases, which helps in modifying the data consistently.

3.2 Input Prompting

Prompting refers to the addition of instructions in the model input, to generate conditional outputs. Jiang et al. (2020) follow a mining based method which follows a relation extraction mechanism followed by a paraphrasing method, which generates identical yet diverse prompts compared to the original prompt. Manually designed rules or a complex selection of input prompts from a discrete space as proposed by Shin et al. (2020) and Gao et al. (2021) can be used. However, selecting prompts from a discrete space would require more training and optimization. For our scenario, where the selection of prompts is done by a separate pipeline, we proceed with the keyphrases extracted from KeyBERT. Our method is similar to Puri et al. (2020), where instead of a question and passage tokens, we have article tokens and event tokens.

3.3 Sentence Embeddings

Word embeddings such as Glove (Pennington et al., 2014) and Word2Vec (Mikolov et al., 2013) provide a vector space for representing words. Word2Vec works on the principle of a context window, where it takes into account a fixed set of previous and next words for modelling its embedding. Glove leverages local and global information for generating word embeddings. Contextualized Embeddings, where a word's embedding depends on the context it is used in, were proposed in BERT (Devlin et al., 2019), which follow a masked language modelling and next sentence prediction tasks, where masked language modelling is a word level task and next sentence prediction is a sentence level task. Sentence-BERT (Reimers and Gurevych, 2019) leverages these contextualized embeddings along with a pooling layer to get a single embedding for the sentence, and a triplet loss function for generating sentence embeddings. However, this method requires having labelled data for positive and negative sentences. Chen et al. (2018) use a generalized pooling method, using a vector based weight multiplication, instead of a simple operation like max, or average. This method is trainable in an end to end fashion, without any additional data requirements. We perform a similar pooling operation on our event embeddings to get event representations which will be used by the decoder.

4 Methodology

4.1 Event Extraction

We use input prompting to guide the summarization task performed by BART (Lewis et al., 2020). This auxiliary input has an event sequence $\mathbf{E} = \mathbf{e_1}, ..., \mathbf{e_b}$ where $\mathbf{b}$ is the number of events and $\mathbf{e_i}$ is an event. Each event consists of $\mathbf{k}$ different tokens. Thus, an event $\mathbf{e_i} = \mathbf{e_{i1}}, ..., \mathbf{e_{ik}}$ where $\mathbf{e_{ij}} \, \varepsilon \, \mathbb{R}^{d_h}$, where d_h is the size of hidden representation. For event extraction, we use KeyBERT (Grootendorst, 2020) to extract keyphrases. KeyBERT uses BERT embeddings to create keywords and keyphrases that have maximum similarity to the document. This similarity is calculated using cosine similarity.

A word overlap threshold $\mathbf{t}$ is set to factor in diversity. Each key phrase is tokenized and padded to reach a fixed-length $\mathbf{k}$. The events are concatenated and inserted before the source text. Thus, input to the BART encoder is a sequence of events followed by the source text.

4.2 Input prompting

To make use of the events extracted from the article, we need to prompt the event data. The tokenized events are added before the tokenized source text. Thus the input to the encoder is $\mathbf{E_{1:b}X_{1:n}}$, where $\mathbf{n}$ is the length of the source text and $\mathbf{b}$ is the number of events. Individual events and event information and source text are separated by separator tokens.

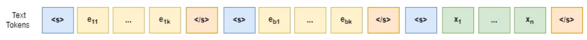


Figure 1: Event information prompted by adding event tokens prior to article tokens.

Event attention masks and source text attention masks are generated to distinguish event information from the source text. This modification is made to the input of the BART architecture to incorporate event information. This kind of concatenation helps us generate event embeddings in a similar manner to source text embeddings.

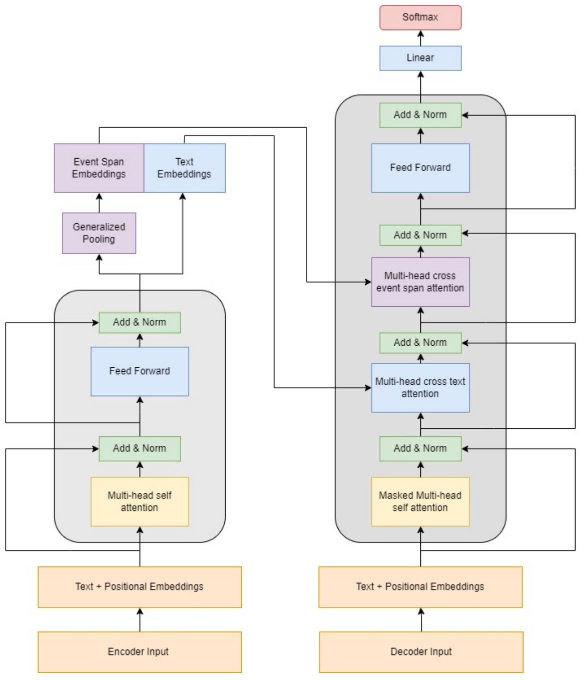
4.3 Summarization model

The summarization model follows a sequence-to-sequence transformer architecture consisting of an encoder and a decoder. We use BART-base as our base model for the architecture. BART-base consists of **N** = 6 layers of encoder and decoder, with the encoder consisting of multihead self-attention block, and the decoder consisting of multihead masked self-attention and cross-attention mechanism. Pooling is performed on the event embeddings to generate event representations. These event embeddings are sent to the decoder. In addition to these attention blocks, we propose the use of an additional attention block, called event attention, which applies cross attention between ground truth summary and the events extracted from our source text. The purpose of this block is to understand how much importance the events hold with respect to the ground truth summary.

4.3.1 Encoder

The encoder for BART-base consists of **N** = 6 layers, each with a self attention mechanism, feed forward layers and residual connections between the layers. The encoder's input is $\mathbf{E_{1:b}X_{1:n}}$, source text attention mask, and event attention mask, where $\mathbf{n}$ is the length of the source text and $\mathbf{b}$ is the number of events. The positional embeddings are added to these tokens and fed into the encoder. Output of each encoder layer is fed into the next encoder layer, for all layers from **L** = 1 to **L** = **N** - 1. Each layer produces embeddings of dimension $\mathbb{R}^{d_h}$. After the layer **L** = **N**, the output of the $\mathbf{N}^{th}$ layer is separated to get event and article embeddings. Generalized pooling is performed on the event embeddings to get a representative embedding for each of the events.

The attention mechanism in the encoder consists of multiple heads. The attention block takes three inputs: queries $\mathbf{Q} \,\varepsilon\, \mathbb{R}^{d_q}$, keys $\mathbf{K} \,\varepsilon\, \mathbb{R}^{d_k}$ and values $\mathbf{V} \,\varepsilon\, \mathbb{R}^{d_v}$ where $\mathbf{d_q}$, $\mathbf{d_k}$, and $\mathbf{d_v}$ are the dimensions of queries, keys and values respectively. A similarity (dot product) between the keys $\mathbf{K}$ and queries $\mathbf{Q}$ is computed followed by the softmax function. Multiplying these scores with values $\mathbf{V}$ gives us the output for the attention block.


Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-head cross event span attention
Add & Norm
Multi-head cross text attention
Add & Norm
Masked Multi-head self attention
Event Span Embeddings
Text Embeddings
Generalized Pooling
Add & Norm
Feed Forward
Add & Norm
Multi-head self attention
Text + Positional Embeddings
Text + Positional Embeddings
Encoder Input
Decoder Input
Figure 2: Model architecture consisting of a transformer encoder and decoder. Generalized pooling is performed on event information. Pooled event embeddings and encoder output is sent to the decoder. Decoder consists of an additional attention block, where attention is computed between events and ground truth summaries.

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

The attention block consists of multiple heads, with each head focusing on a different representation. Outputs of each of these heads are concatenated and multiplied by a weight matrix to get the output for the attention block.

The queries, keys and values come from the encoder input for **L** = 1. For layers, **L** = 2 to **N**, queries, keys and values come from the previous encoder layers. Encoder self-attention is not masked hence, can attend to all the positions of the input.

4.3.2 Generalized Pooling

To get a representation for each event, we employ a method to go from token embeddings to an embedding for each event. We split the output of the final encoder layer into event embeddings and source text embeddings. We use a weighted pooling mechanism to form a single vector representation of each event. However, we use a one-dimensional convolutional layer, instead of a feed forward layer (Chen et al., 2018). The idea behind using a convolutional layer is to get a sliding window so that each event can be handled without recursively passing it through the feed-forward layer. For an input $\mathbf{X_{1:n}}$ having events $\mathbf{E} = \mathbf{e_1}, \ldots, \mathbf{e_b}$ where $\mathbf{e_i} = \mathbf{e_{i1}}, ..., \mathbf{e_{ik}}$, a single representation is calculated for every event. $\mathbf{PE}$ represents the weighted pooled embeddings. Thus, the input to the decoder is $\mathbf{PE_{1:b}X_{1:n}}$ where $\mathbf{PE} = \mathbf{pe_1}, ..., \mathbf{pe_b}$.

$$pe_i = Conv1D(e_i) \tag{2}$$

4.3.3 Decoder

The decoder for BART-base consists of $\mathbf{N}$ = 6 layers, each with a masked self-attention mechanism, masked cross attention and a masked event attention block. The input to the decoder is $\mathbf{Y_{1:p}}$ as ground truth summary and $\mathbf{PE_{1:b}X_{1:n}}$ as encoder output, where $\mathbf{p}$ is the length of the ground truth summary. The output of each decoder layer is fed into the next decoder layer, for all layers from $\mathbf{L}$ = 1 to $\mathbf{L} = \mathbf{N} - \mathbf{1}$. Each layer produces embeddings of dimension $\mathbf{d_h}$. After the la yer $\mathbf{L} = \mathbf{N}$, the output is passed through a feed-forward layer and softmax activation over the vocabulary to get the logits which are used for generating the summary.

The attention mechanism in the decoder is masked so that the decoder does not attend to inputs of time steps ahead of the current time step. These attention blocks are similar to the attention block in the encoder. The self-attention block uses ground truth summary as query, key, and value. For the cross attention mechanism, $\mathbf{X_{1:n}}$ is used for keys and values, whereas queries are obtained from the previous decoder layer. Similarly, for the event attention block, the queries are obtained from the previous decoder layer and the keys and values are obtained from the pooled event embeddings $\mathbf{PE_{1:b}}$.

5 Experimental Setup

5.1 Dataset

We have used CNN / Daily Mail (Nallapati et al., 2016) and XSUM (Narayan et al., 2018) datasets for our experimentation. CNN / Daily Mail consists of news articles and their abstractive summaries.

	CNN / Daily Mail	**XSUM**
`Train`	287113	204045
`Validation`	13368	11332
`Test`	11490	11334

Table 2: Number of data points in training, validation and testing sets for each of the datasets.

XSUM consists of BBC articles which cover a wide variety of domains. Since we are using event-based summarization, we need to first extract the events and add them to the model input along with the article. In our experiments, we have extracted 4 events, each of these being keyphrases consisting of 4 words. We use BART tokenizer to tokenize the dataset. This may result in some words being split into multiple tokens. Thus every event is allocated 10 tokens including the start and end tokens. Since the input size for the encoder is 512, the articles are truncated to 472 tokens to accommodate the 40 tokens for the event sequence. Allocating more tokens to an event or increasing the number of events would decrease the number of tokens that can be taken from the source article. This results in event information vs article length trade-off. For CNN / Daily Mail the ground truth summaries are truncated to 128 tokens, and for XSUM the ground truth summaries are truncated to 64 tokens. This difference between ground truth summary lengths is because, in CNN / Daily Mail, the summaries are highlights from the news article, but in XSUM the summaries are mostly a sentence long. The different splits for the above-mentioned datasets are specified in Table 2. The number of events extracted from some of the articles is less than 4. In such cases, padding is added to reach the 40 tokens allocated for event information.

5.2 Implementation Details

We chose BART-base as our base model, instead of BART-large, due to insufficient resources for training the larger variant of the model. The encoder and decoder each consist of $\mathbf{N}$ = 6 layers. As BART is pre-trained, these pre-trained weights are taken to be the initial model weights. The weights for generalized pooling and event attention blocks are randomly initialized. BART-base consists of 12 attention heads for encoder and decoder, with hidden dimension size $\mathbf{d_h}$ = 768, and a vocabulary size $\mathbf{V}$ = 50,265. For the generalized pooling block, kernel size $\mathbf{k}$ = 10, a stride of 10, and the number

Model	R1	R2	RL	RLSum	BertScore
Event prompted BART-base	41.57	19.89	29.52	**38.88**	63.79
BART Large-CNN	44.16	21.28	40.90	36.42	**64.14**
Pegasus Large-CNN	**44.17**	**21.47**	**41.11**	36.39	62.52

Table 3: Metric values for CNN / Daily Mail Dataset

Model	R1	R2	RL	RLSum	BertScore
Event prompted BART-base	40.51	18.64	33.27	33.26	66.51
BART Large CNN	45.14	22.27	37.25	36.47	68.64
Pegasus Large CNN	**47.21**	**24.56**	**39.25**	38.63	**69.99**

Table 4: Metric values for XSUM Dataset

of input and output channels as 768 are used.

We use a cross-entropy loss objective for training the model. Our model has a total of 159M parameters, 139M parameters due to BART-base with an additional 20M parameters due to generalized pooling block and event attention block. We use NVIDIA Quadro RTX 6000 16GB GPU for running experiments on the model. We use a batch size of 8 for training and validation. The learning rate is set to 1e-05 with a linear learning rate scheduler for CNN / Daily Mail and a polynomial learning rate scheduler for XSUM. Weight decay is set to 5e-04 for CNN / Daily Mail and 1e-04 for XSUM and the models are trained to 500k steps for CNN / Daily Mail and 250k steps for XSUM. While decoding, we use beam search with a beam size of 5 for both datasets.

6 Results and Analysis

6.1 Metric Evaluation

The results of our model are quantified using Rouge1, Rouge2, RougeL, RougeLSum and BERTscore (Zhang* et al., 2020). Rouge1 and Rouge2 measure the uni-gram and bi-gram matches respectively. RougeL measures the longest common subsequence. RougeLSum is a variation of RougeL and it differs from RougeL in the treatment of the newline character. BERTscore computes the semantic similarity between generated and ground truth summary. We observe in Table 3 and Table 4 that rouge scores from our model are comparable with BART-Large and Pegasus-Large.

Since our summaries revolve around events, we compute the rouge scores between the identified events vs ground truth and identified events vs generated summaries. This will showcase the overlap between the events and their respective generated summaries. We randomly select 25 summaries from each of the datasets and compute rouge scores between each of the events vs the ground truth, and each of the events vs the summaries generated by

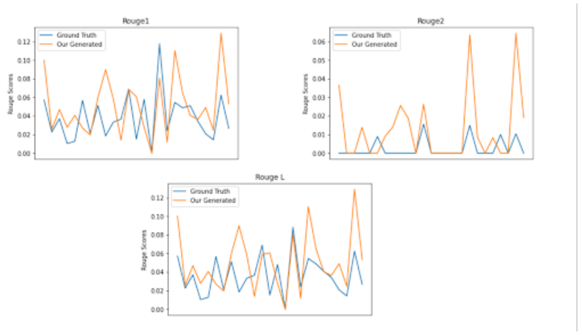


Figure 3: Rouge Scores between events vs ground truth, and events vs summaries generated by Event prompted BART-base for CNN / Daily Mail.

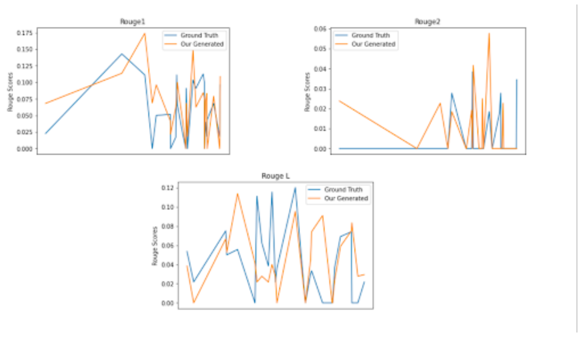


Figure 4: Rouge Scores between events vs ground truth, and events vs summaries generated by Event prompted BART-base for XSUM.

our model. We average out the scores returned between a summary and each of the identified events to get a single value. We observe in Figure 3 and Figure 4, that the rouge scores between events and summaries generated by our model is close to or higher than the rouge scores between events and ground truth summaries.

CNN / Daily Mail and XSUM are both datasets used for abstractive summarization, however, the expectations from the generated summaries are different in both cases. CNN / Daily Mail consists of longer ground truth summaries which explain the article in a few sentences. On the other hand, XSUM consists of significantly shorter ground truth summaries. As observed in Figure 5 and Figure 6, the length of summaries generated by our model is similar to the length of ground truth summaries. While calculating the length of text, we tokenize the text using BART tokenizer and consider the number of tokens output by the BART tokenizer as the length of the text. In Figure 7 and Figure 8 the lengths of the generated summaries are divided into groups of 10 and the average rouge scores for all the summaries in a group is calculated.

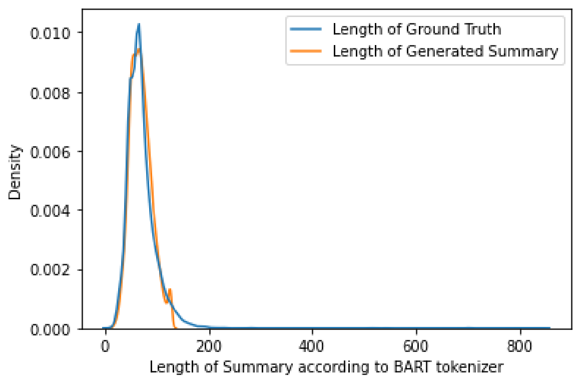


Figure 5: Distribution of summary lengths as calculated by the number of tokens generated by the BART tokenizer for the CNN / Daily Mail dataset.

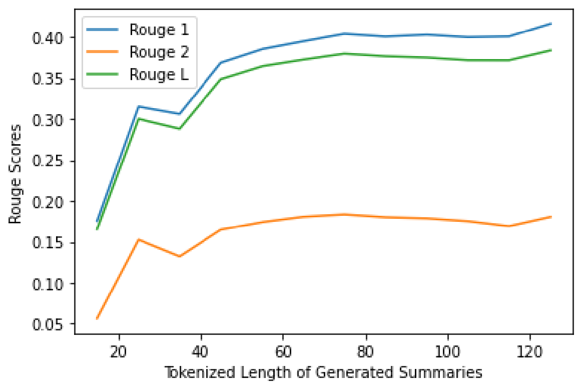


Figure 7: The average rouge scores for all the summaries in a group for CNN / Daily Mail.

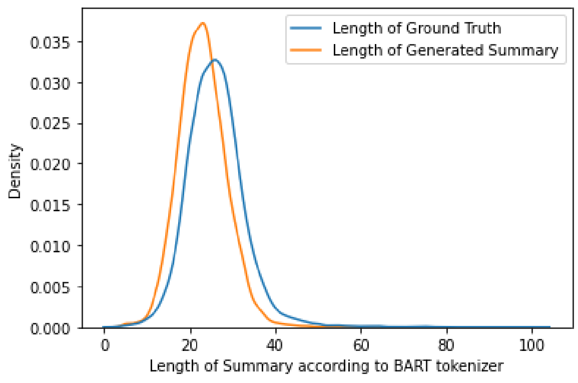


Figure 6: Distribution of summary lengths as calculated by the number of tokens generated by the BART tokenizer for the XSUM dataset.

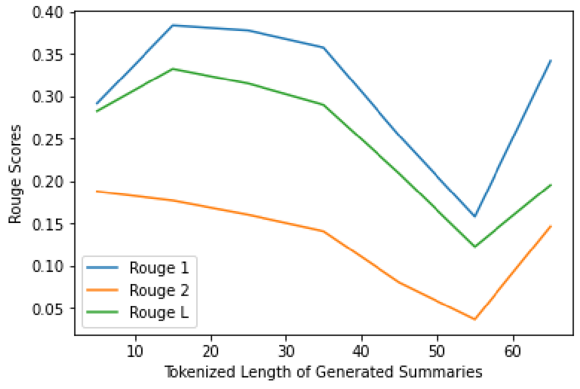


Figure 8: The average rouge scores for all the summaries in a group for XSUM.

It can be observed in Figure 7 that Rouge scores increase as the length of the articles increases in the CNN / Daily Mail dataset. However, in Figure 8 we observe that for the XSUM dataset, longer summaries seem to have lower rouge scores.

6.2 Human Evaluation

To capture the subjectiveness and diversity of language generation tasks such as summarization, we conduct human evaluation. Since we have added event input prompting to our model, we aim to utilize human evaluation as a method to understand to what extent summaries generated by our model are influenced by the events. Three evaluators, fluent in the English language were sent summaries generated by BART-Large-CNN, PEGASUS-Large-CNN and Event prompted BART-base, along with their respective source text, extracted events and ground truth summaries. The names of the models that generated the summaries to be evaluated, were not shared with the evaluators. The evaluators were provided with 25 such data points from the test split and a list of metrics to grade the summaries on. The metrics were fluency, event inclusiveness, factual correctness, coherence, and informativeness. Fluency is used to verify if the text generated has the correct grammatical structure and rules. Since our model incorporates input prompting using events, we use event inclusiveness as a metric to capture how much the summaries are influenced by the identified events. Factual correctness is included to confirm if the facts in the summary are consistent with the facts in the source text and ground truth. For understanding to what degree the summary makes sense as a whole, coherence is added as a metric. Informativeness is used to verify if the most important points of the article are present in the summary.

The average of the scores was taken across all the data points for different models and their metrics. The average fluency, event inclusiveness, factual correctness, coherence, and informativeness scores for BART-Large-CNN, BART-base prompted by

Metrics		Event prompted BART-base	BART Large-CNN	Pegasus Large-CNN
Fluency	P1	4.12	**4.32**	3.76
	P2	4.88	**4.92**	4.72
	P3	4.64	**4.76**	4.40
Event inclusivness	P1	3.92	**4.16**	3.60
	P2	**3.84**	2.56	2.04
	P3	**4.16**	3.72	3.68
Factual Correctness	P1	**4.08**	4.04	3.80
	P2	**4.96**	4.88	4.56
	P3	**4.76**	4.64	4.44
Coherence	P1	**3.80**	3.60	3.28
	P2	**4.24**	3.84	3.88
	P3	**4.44**	4.40	3.64
Informativness	P1	3.52	**3.56**	3.16
	P2	**3.44**	**3.44**	2.68
	P3	3.80	**3.84**	3.20

Table 5: Human Evaluation results for the CNN / Daily Mail dataset.

events, and PEGASUS-Large CNN are shown in Table 5, where P1, P2, and P3 refer to the the three evaluators.

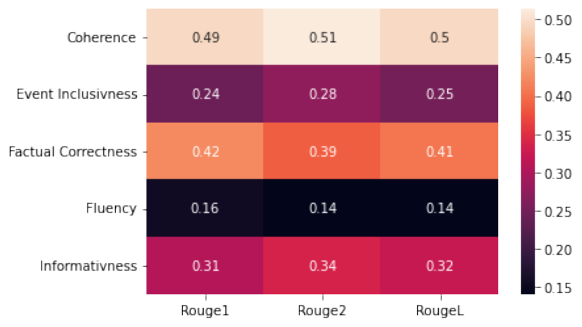


Figure 9: Correlation matrix between the rouge scores and human evaluation metrics. Scores assigned by reviewers are averaged out.

We can observe in that our scores for fluency are comparable to the scores for BART Large-CNN. We can see that 2 among 3 reviewers rated our generated summaries as being the most event inclusive. The first reviewer gave our summaries a score very close to the BART-Large score. For factual correctness and coherence the scores are highest for our generated summaries. For informativness, we observe that our scores are very similar to BART-Large scores, which rank the highest in the informativeness category.

7 Conclusion and Future Work

We introduce event prompting and an additional event attention block in the existing BART-base architecture to enable the model to generate summaries related to the events identified in the source text. Our model achieves comparable Rouge and BERT scores as the larger versions of BART and PEGASUS (Zhang et al., 2020). We also carry out human evaluation for our trained model, and achieve higher scores for event information inclusiveness as compared to the other transformer based models.

8 Acknowledgements

We thank Mr. Mayank Chowdhary, Mr. Jasdeep Singh Grover, Mr. Tim Fischer and the reviewers for their invaluable feedback.

References

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, et al. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the ninth workshop on statistical machine translation*, pages 12–58.

Qian Chen, Zhen-Hua Ling, and Xiaodan Zhu. 2018. Enhancing sentence embedding with generalized pooling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1815–1826, Santa Fe, New Mexico, USA. 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.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. 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 3816–3830, Online. Association for Computational Linguistics.

Maarten Grootendorst. 2020. Keybert: Minimal keyword extraction with bert.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

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*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 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*, pages 7871–7880, Online. Association for Computational Linguistics.

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.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

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

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

Faik Kerem Örs, Süveyda Yeniterzi, and Reyyan Yeniterzi. 2020. Event clustering within news articles. In *Proceedings of the Workshop on Automated Extraction of Socio-political Events from News 2020*, pages 63–68.

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.

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.

Raul Puri, Ryan Spring, Mohammad Shoeybi, Mostofa Patwary, and Bryan Catanzaro. 2020. Training question answering models from synthetic data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5811–5826, Online. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

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 and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Alan Ritter, Oren Etzioni, and Sam Clark. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1104–1112.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Si Sun, Zhenghao Liu, Chenyan Xiong, Zhiyuan Liu, and Jie Bao. 2021. Capturing global informativeness in open domain keyphrase extraction. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 275–287. Springer.

Marco A Valenzuela-Escárcega, Gus Hahn-Powell, Mihai Surdeanu, and Thomas Hicks. 2015. A domain-independent rule-based framework for event extraction. In *Proceedings of ACL-IJCNLP 2015 System Demonstrations*, pages 127–132.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Runxin Xu, Tianyu Liu, Lei Li, and Baobao Chang. 2021. Document-level event extraction via heterogeneous graph-based interaction model with a tracker. 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 3533–3546, Online. Association for Computational Linguistics.

Haoyu Zhang, Jingjing Cai, Jianjun Xu, and Ji Wang. 2019. Pretraining-based natural language generation for text summarization. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 789–797, Hong Kong, China. Association for Computational Linguistics.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Augmenting ebooks with Recommended Questions using Contrastive fine-tuned T5

Shobhan Kumar
IIIT Dharwad
Karnataka, India
shobhank9@gmail.com

Arun Chauhan
IIIT Dharwad
Karnataka, India
aruntakhur@gmail.com

Pavan Kumar C
IIIT Dharwad
Karnataka, India
pavan@iiitdwd.ac.in

Abstract

The recent advances in Artificial Intelligence (AI) has made generation of questions from natural language text possible, this approach completely excludes human in the loop, while generating the appropriate questions which improves the students learning engagement. The ever growing rate of educational content renders it increasingly difficult to manually generate sufficient practice or quiz questions to accompany it. Reading comprehension can be improved by asking the right questions. In this work a transformer based question generation model specifically made for autonomously producing quiz questions from educational information, such as ebooks is introduced. This work proposes an contrastive training approach for “Text-to-Text Transfer Transformer" (T5) model where the model (T5-eQG) creates the summarised text for the input document and then automatically generates the questions. Our model shows promising results over earlier Neural Network based and rules based models for question generating task on benchmark datasets and NCERT ebooks.

1 Introduction

Textbooks are a primary source of information for students (Kumar and Chauhan, 2022). Besides this students tends to study ebooks, lecture notes, and MOOCs for further knowledge acquisition (Herrera et al., 2018; Kumar and Chauhan, 2020). With these reading materials students can only partially understand the material presented, and it does not makes their learning effective (Ebersbach et al., 2020). Asking questions about the reading content and evaluate the answer is a intuitive way of promoting learning (Xu et al., 2021; Ruan et al., 2019). This motivates us to look into ways to generate educational questions for ebook content to aid the students learning.

It is difficult to define the specific process for asking insightful educational questions about texts, which entails doing more than just writing fluid, natural-sounding texts. It is bit hard to generate relevant educational questions on text (Horbach et al., 2020). Typically, it involves gathering important instructional facts and turning them into questions. Few attempts have been made in the recent years to complete this task by using statistical and Neural Network based algorithms to choose crucial passages (Chen et al., 2019; Du et al., 2017) and concepts and produce insightful questions (Dong et al., 2019; Steuer et al., 2020). Education question generation on text, however, has not received much attention.

The proposed educational Question Generation (eQG) model presents a set of questions for each chapter of the ebook. Teachers might use these questions for self-study, before discussing the subject in class, which would helps them for deep knowledge transfer to the students (Kumar and Chauhan, 2019; Xu et al., 2021). We assess our methodology on QA dataset (HotPot QA(Yang et al., 2018), FairytaleQA[1]) as well as PRML (Bishop, 2006) and NCERT[2] eBooks. Table 1 presents sample results of our eQG model for the input text (refer Figure 1). The learner can use eQGs for self-assessment questions to gauge their conceptual understanding.

The contributions of this work are two fold:

- Text summarizer: We fine-tune the Text-to-Text transformer (T5) to extract the informative sentences that are most likely for educators to design questions for the original input.
- Contrastive training for T5-eQG: We fine-tune the T5 transformer on positive and negative training samples. A contrastive loss is added between the positive and negative training feature pairs during the fine-tuning process. It

[1]https://github.com/uci-soe/FairytaleQAData
[2]https://ncert.nic.in/textbook.php

helps in generating more complex questions on input document.

2 Background

Natural language creation has primarily evolved through statistical learning in recent years. To create manuscripts that resemble human writing, the models imitate linguistic conventions. In recent years, the natural language processing communities have shown a great deal of interest in the question generation (QG) task(Wang et al., 2017; Lyu et al., 2021), which creates a natural question corresponding to the supplied text or answer phase. The syntactic cues have been used in the rule based model to create Questions (De Kuthy et al., 2020). A back translation tool was paired with a syntactic question generator to eliminate grammatical errors and increase robustness(Dhole and Manning, 2020). Declarative sentences were transformed into natural questions by the emergence of sequence-to-sequence models (Radford et al., 2019). Applying pre-trained transformers or various optimization objectives (Qi et al., 2020) led to further advancements. Previous research has explored the importance of QG model in teaching learning process(Kurdi et al., 2020).

For QA and question generation, NarrativeQA (Kočiský et al., 2018) aims to incorporate important information from many places inside a paragraph. Similar to this, the MS MARCO (Nguyen et al., 2016) dataset combines many sources of responses to search queries. The employment of a reinforcement learning agent to align questions from various documents is proposed as a contrastive strategy, where supervised model is trained to produce questions on a text (Cho et al., 2021). To achieve good performance, questions with summaries and reports were generated using a rule-based methodology (Lyu et al., 2021). The solutions discussed above typically don't take the educational component into account and could not be effective in the real world edu QG task. Our research focuses on the generating question on e-book content, in this work we use FairytaleQA dataset (Xu et al., 2022). For each paragraph in FairytaleQA, experts typically create a different style of question. We propose that context is a key factor in determining the kinds of questions that ought to be made while reading e-books.

3 Methodology

Figure 2 depicts the overall architecture of our eQG system, which consists of two modules: i) Text summarizer ii) Question-generation(QG). We first create summaries of type s with the input paragraph d, and then generate the questions q on summarized text. The generated questions are said to be relevant if the question q_i can be answered with the paragraph d_i and this is formulated as maximizing the conditional probability $p(q|d)$:

$$q = argmax(p(q \mid d)) = argmax \prod_{i=1}^{L} p(w_i \mid d, q_{i'}) \tag{1}$$

where w_i is the i^{th} token of the generated question q, and $q_{i'}$ denotes the previous decoded tokens, i.e., $q_1, \ldots, q_{i\text{-}1}$.

T5 - Abstractive summarizer: In this work, we examined the text summarizer and QG as a task of text-to-text transformation. So, we first train a T5[3] summarising model to produce the abstract summary of the input text.

Edu Question Generation: Once the model produce the abstract summary of the input text, next step is to generate an educational question out of it. We train a T5-QG model directly on top of the summary using the annotated questions, since T5-summary model already has knowledge on rich informative text.

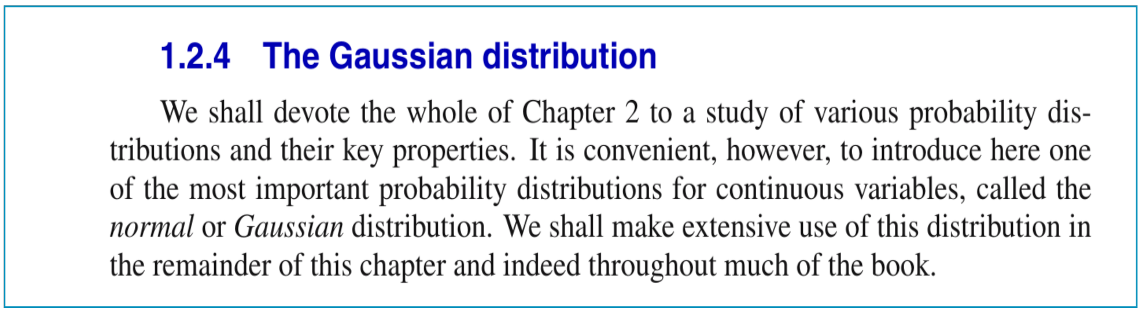
When the model is fine tuned with tiny dataset, fine-tuning process with QA task loss is generally insufficient to achieve satisfactory performance. To address this issue, we generated the negative samples for each document d_i and fine-tune using both the data samples.

Most of the existing QG model suffers from the exposure bias problem. Therefore, we created a negative sample and trained an end-to-end eQG model by introducing contrastive loss function. We trained two variants of T5_QG model. At first we fine-tune the T5_QG model by minimizing the cross-entropy loss. In the next step, model is trained on augmented data (both ground truth question and generated negative samples) with the contrastive loss and cross-entropy loss

$$T_{loss} = Q_{\text{task}} + Q_{\text{c\_loss}} \tag{2}$$

where T_{loss} is the total loss, Q_{task} and $Q_{\text{c\_loss}}$ are the QG task loss and the contrastive loss, respectively,

[3]https://huggingface.co/docs/transformers/model_doc/t5



1.2.4 The Gaussian distribution

We shall devote the whole of Chapter 2 to a study of various probability distributions and their key properties. It is convenient, however, to introduce here one of the most important probability distributions for continuous variables, called the *normal* or *Gaussian* distribution. We shall make extensive use of this distribution in the remainder of this chapter and indeed throughout much of the book.

Figure 1: The sample input text from PRML ebook (Section 1.2) by C.Bishop(Bishop, 2006)

Table 1: The generated questions using our eQG model (input text-PRML ebook).

Text: Refer Figure 1	
Abstract summary: The Gaussian distribution It is convenient, however, to introduce here one of the most important probability distributions for continuous variables	
Generated top k questions using eQG model (k=3)	
Q1	What is Gaussian distribution?
Q2	What is the most important probability distribution for continuous variables?
Q3	Where we use normal distribution?

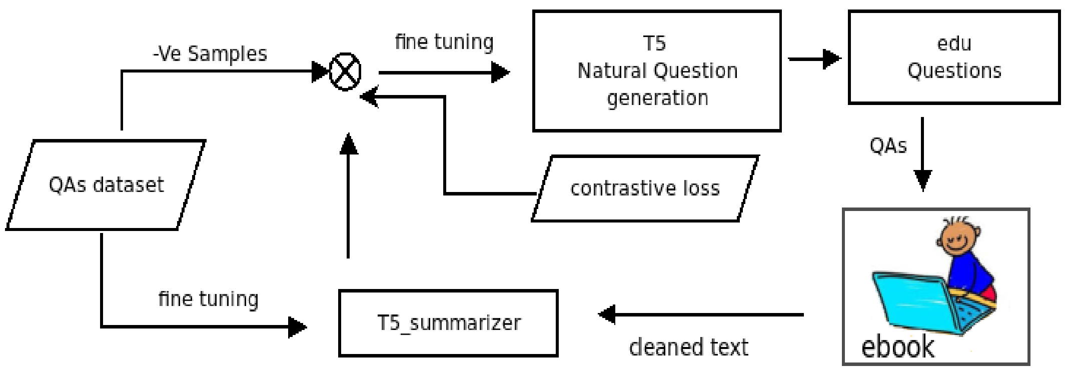


Figure 2: Overall architecture of our eQG model.

$$Q_{\text{c\_loss}} = q_l * D_w^2 + (1-q_l) * max(m-d_w, 0)^2 \quad (3)$$

where q_l is the ground-truth labels from our dataset, d_w is the Euclidean distance and m is the margin used for the contrastive loss function.

4 Experimental Results

In this work we trained T5 (Text-to-Text Transfer Transformer) base model from hugging face transformers[3]. An input sequence and a corresponding target sequence are required for every training run of T5. The model receives the input sequence via input ids. The target sequence is provided to the decoder using the decoder input ids after being prepended by a start-sequence token and moved to the right. The EOS token is subsequently attached to the target sequence in teacher-forcing fashion, which correlates to the labels. The start-sequence token here is the PAD token.

Data set: We used FairytaleQA[1] and HotpotQA data, the FairytaleQA has 10,580 QA-pairs, which were drawn from 278 different novels. The HotpotQA(Yang et al., 2018) has ≈100K QA pairings on Wikipedia articles. For FairyQA we divide the data into 8.5K/1K/1K, and for HotPot QA 84,512, 6K and 6K samples as train,validation and test data.

For fine tuning the T-5[3] model we used the AdaFactor optimizer and a maximum sequence length is set to 512, model is trained for 4 epochs. We follow the grid-search approach for choosing the best set of training parameters (learning-rate:{{2,3,5}e-3,4}} and batch size: {8,16,32,64}, warm-up ratio: {0.1}). During the experiment, we found that a mini-batch size of 32 (learning rate: 3e-3) produces acceptable results.

Table 1 and Table 2 highlight the generated questions for text from PRML (Bishop, 2006) and NCERT[2] CCT ebook. The generated questions makes the students learning more effective, and assist them in improving their conceptual understanding ability.

We validated the quality of questions generated on three experimental configuration. QG model trained i)only on HotpotQA, (ii) only on FairytaleQA, iii) on both FairytaleQA and HotpotQA. The third setting shows a significant improvement over the preceding setups, so this was chosen as our final QG model for further comparison to the earlier work. Results are shown in Table 3. We see that the model optimised on FairytaleQA alone shows significant improvement over the model trained on both the dataset. This is due to the disparities in domain and distribution between the two datasets. The third settings shows a decent results during the state-of-the-art model comparison.

Table 4 provides the comparison results of our model with state-of-the-art models. Our model achieves a comparable BLEU4, Rouge scores with the cutting-edge QG model in HotpotQA without using the answer information or any external linguistic knowledge. This illustrates the effectiveness of contrastive training of language model for the QG task.

5 Conclusion and Future scope

The use of AI is constantly evolving in diverse applications. This study investigates the potential advantages of a natural language processing approach for education. This research presented an education question generating (eQG) approach that augments the ebook content with generated edu-questions to provide students with an effective learning platform. Through experiments, we assessed the model's performance on a question generation task both before and after contrastive training. We discovered that a contrastive trained model can produce more pertinent questions on the input text and can comprehend key concepts more effectively. Experiments on QA dataset, PRML(Bishop, 2006) and NCERT [2] ebook shows that our model succeeds to produces complex questions at scale.

The possible future direction could be

- Design a context-aware QG model, where the generation of a new text is conditioned on previous generations as well as the ebook contents.
- Conduct a human evaluation to validate the appropriateness of the generated questions on ebook content.

Table 2: The abstract summary and generated questions using our eQG model(top k questions) for the input text NCERT[2] CCT ebook.

Input Text:Use of innovative technologies like Silicon-On-Insulator (SOI), Complementary Metal-Oxide-Semiconductor (CMOS), capacitor -less memory, Micro-Optic-Electro-Mechanical-System (MOEMS) III-V compound materials-on-insulator and others have improved the performance and also reduced the size of consumer electronic devices...	
Abstract summary: Innovative technologies such as Silicon-On-Insulator (SOI), Complementary Metal-Oxide-Semiconductor (CMOS), capacitor-less memory, Micro-Optic...	
Generated Top k (k=3) questions	
Q1	What are the benefits of using silicon-on-insulator (SOI)?
Q2	How is graphene expected to improve the processing speed of computers?
Q3	What is the advantage of using III-V compound materials-on-insulator in consumer electronic devices?

Table 3: Comparison of our contrastive eQG models with various experimental settings.

QG model	Evaluation metric:Rouge-L	
	Validation_data	Test_data
$T5_{base}$_HotpotQA	0.423	0.441
$T5_{base}$_FairytaleQA	0.512	0.526
$T5_{base}$_HotpotQA_FairytaleQA	0.507	0.518

Table 4: The comparison results of our model with prior work for QG task on HotPot dataset.

QG model	Evaluation metrics			
	BLEU-1	BLEU-4	Meteor	Rouge-L
RL_QG (Xie et al., 2020)	37.97	15.41	19.61	35.12
Deep_QG(Pan et al., 2020)	40.55	15.53	20.15	36.94
T5QG	40.96	17.54	19.21	42.36
Contrastive_T5QG	42.04	19.11	20.07	48.50

References

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

Guanliang Chen, Jie Yang, and Dragan Gasevic. 2019. A comparative study on question-worthy sentence selection strategies for educational question generation. In *Artificial Intelligence in Education*, pages 59–70, Cham. Springer International Publishing.

Woon Sang Cho, Yizhe Zhang, Sudha Rao, Asli Celikyilmaz, Chenyan Xiong, Jianfeng Gao, Mengdi Wang, and Bill Dolan. 2021. Contrastive multi-document qg. In *Proc.16th European Conference, Chapter of the ACL*, pages 12–30.

Kordula De Kuthy, Madeeswaran Kannan, Haemanth Santhi Ponnusamy, and Detmar Meurers. 2020. Towards automatically generating questions under discussion to link information and discourse structure. In *Proc.28th International Conference on Computational Linguistics*, pages 5786–5798.

Kaustubh D. Dhole and Christopher D. Manning. 2020. Syn-qg: Syntactic and shallow semantic rules for question generation. *ArXiv*, abs/2004.08694.

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*. Curran Associates Inc., Red Hook, NY, USA.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. *CoRR*, abs/1705.00106.

Mirjam Ebersbach, Maike Feierabend, and Katharina Barzagar B. Nazari. 2020. Comparing the effects of generating questions, testing, and restudying on students' long-term recall in university learning. *Applied Cognitive Psychology*, 34(3):724–736.

Jose Miguel Herrera, Barbara Poblete, and Denis Parra. 2018. Learning to leverage microblog information for qa retrieval. In *ECIR*, volume 10772 of *Lecture Notes in Computer Science*, pages 507–520. Springer.

Andrea Horbach, Itziar Aldabe, Marie Bexte, Oier Lopez de Lacalle, and Montse Maritxalar. 2020. Linguistic appropriateness and pedagogic usefulness of reading comprehension questions. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 1753–1762, Marseille, France. European Language Resources Association.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Shobhan Kumar and Arun Chauhan. 2019. Enriching textbooks by question-answers using cqa. In *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*, pages 707–714.

Shobhan Kumar and Arun Chauhan. 2020. Recommending question-answers for enriching textbooks. In *Big Data Analytics*, pages 308–328, Cham. Springer International Publishing.

Shobhan Kumar and Arun Chauhan. 2022. Augmenting textbooks with cqa question-answers and annotated youtube videos to increase its relevance. *Neural Process Lett*.

Ghader Kurdi, Jared Leo, Bijan Parsia, Uli Sattler, and Salam Al-Emari. 2020. A systematic review of automatic question generation for educational purposes. *International Journal of Artificial Intelligence in Education*, 30:121–204.

Chenyang Lyu, Lifeng Shang, Yvette Graham, Jennifer Foster, Xin Jiang, and Qun Liu. 2021. Improving unsupervised question answering via summarization-informed question generation. In *Proc.EMNLP*, pages 4134–4148.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. *CoRR*, abs/1611.09268.

Liangming Pan, Yuxi Xie, Yansong Feng, Tat-Seng Chua, and Min-Yen Kan. 2020. Semantic graphs for generating deep questions. In *Proc. 58th Annual Meeting of the ACL*, pages 1463–1475, Online.

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-SequencePre-training. In *EMNLP 2020*, pages 2401–2410.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.

Sherry Ruan, Liwei Jiang, Justin Xu, Bryce Joe-Kun Tham, Zhengneng Qiu, Yeshuang Zhu, Elizabeth L. Murnane, Emma Brunskill, and James A. Landay. 2019. Quizbot: A dialogue-based adaptive learning system for factual knowledge. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–13, New York, NY, USA. Association for Computing Machinery.

Tim Steuer, Anna Filighera, and Christoph Rensing. 2020. Remember the facts? investigating answer-aware neural question generation for text comprehension. In *Artificial Intelligence in Education*, pages 512–523, Cham. Springer International Publishing.

Tong Wang, Xingdi (Eric) Yuan, and Adam Trischler. 2017. A joint model for question answering and question generation. In *Learning to generate natural language workshop, ICML 2017*.

Yuxi Xie, Liangming Pan, Dongzhe Wang, Min-Yen Kan, and Yansong Feng. 2020. Exploring question-specific rewards for generating deep questions. In *Proc. 28th International Conference on Computational Linguistics*, pages 2534–2546.

Ying Xu, Dakuo Wang, Penelope Collins, Hyelim Lee, and Mark Warschauer. 2021. Same benefits, different communication patterns: Comparing children's reading with a conversational agent vs. a human partner. *Computers and Education*.

Ying Xu, Dakuo Wang, Mo Yu, Daniel Ritchie, Bingsheng Yao, Tongshuang Wu, Zheng Zhang, Toby Jia-Jun Li, Nora Bradford, Branda Sun, Tran Bao Hoang, Yisi Sang, Yufang Hou, Xiaojuan Ma, Diyi Yang, Nanyun Peng, Zhou Yu, and Mark Warschauer. 2022. Fantastic questions and where to find them: FairytaleQA – an authentic dataset for narrative comprehension. Association for Computational Linguistics.

Zhilin Yang, Peng, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *CoRR*, abs/1809.09600.

Reducing Inference Time of Biomedical NER Tasks using Multi-Task Learning

Mukund Chaudhry[1] **Arman Kazmi**[1] **Akhilesh Verma**[1] **Vishal Samal** [1]
Shashank Jatav[1] **Kristopher Paul**[1] **Ashutosh Modi**[2]
Elucidata Inc., New Delhi[1] IIT Kanpur[2]
ashutoshm@cse.iitk.ac.in[2]
{mukund.chaudhry, arman.kazmi, akhilesh.verma, vishal.samal,
shashank.jatav, kristopher.paul } @elucidata.io[1]

Abstract

Recently, fine-tuned transformer-based models (e.g., PubMedBERT, BioBERT) have shown the state-of-the-art performance of several BioNLP tasks, such as Named Entity Recognition (NER). However, transformer-based models are complex, have millions of parameters, and are relatively slow during inference. In this paper, we address the time complexity limitations of the BioNLP transformer models. In particular, we propose a Multi-Task Learning based framework for jointly learning three different biomedical NER tasks. Our experiments show a reduction in inference time by a factor of three without any reduction in prediction accuracy.

1 Introduction

Transformer-based large language models (LLMs) have made it much easier to perform various NLP tasks with high accuracy. However, due to their large size, they take a lot of time and money to perform inference on large datasets. To give some perspective, one forward pass through PubMedBERT (Gu et al., 2020) takes 8-50ms on an AWS g4dn.xlarge instance [1] (which comes with an NVIDIA T4 GPU). Running one PubMedBERT model on 1 million biomedical paragraphs would take roughly 9 hours. Given the deluge of biological information daily, using fine-tuned PubMedBERT models for each biomedical NER task separately would be too time-consuming and expensive.

When it comes to deep learning models, there are generally two variables that are optimized before deployment. These are *size* (the space occupied by the model's weight on disk and RAM) and *inference time* (the time taken for one prediction).

Model size tends to matter more when deployed on edge devices and mobile phones since these devices have storage and RAM constraints. Several techniques, such as knowledge distillation, have been proposed to address this issue, and some of the prominent models which have achieved a significant decrease in model size without much decrease in accuracy are DistilBERT (Sanh et al., 2019), SqueezeBERT (Iandola et al., 2020), and MobileBERT (Sun et al., 2020). However, size is usually not an issue for models running on servers. For example, a PubMedBERT model has a size of only 400 MB. Instead, the main concern is inference time, which is what we focus on in this paper. Generally, a reduction in the model size naturally leads to a reduction in the inference time. However, in this work, we focus on reducing the inference time without reducing the model size.

Multi-Task Learning (MTL) primarily aims to improve the accuracy of multiple prediction tasks that are related to each other by leveraging commonly useful information. Many of the previous works have shown the effectiveness of multi-task learning-based models for BioNer tasks. The first work to apply MTL for biomedical named entities was attempted by Crichton et al. (2017). They used pre-trained word embeddings with CNN-based neural networks to extract named entities from biomedical texts. Wang et al. (2018) used a combination of BiLSTM and CRF-based model, adapted from Liu et al. (2018), to extract the entities and further used character and word-based embeddings that were shared by different datasets. A slightly different approach was proposed by Zuo and Zhang (2020), where they trained a dataset-aware MTL model and showed that their model was able to discriminatively exploit information from all of the related training datasets.

The recent developments of large language models, such as BERT (Devlin et al., 2018), have demonstrated the effectiveness of better contextualized representation of various NLP tasks. Lee et al. (2019) developed BioBERT using the BERT language model and pre-trained it on biomedical abstracts and papers. They achieved state-of-the-art

[1] https://aws.amazon.com/ec2/instance-types/g4/

results on several biomedical named entity recognition datasets. Khan et al. (2020) and Mehmood et al. (2019) incorporated MTL in BERT-based models and showed promising results to extract biomedical named entities.

Although the previous works have shown the importance of multi-task learning when incorporated with either neural network-based models or transformer-based models, none of them have targeted optimizing these large models. While deploying these models for prediction, inference time matters; hence, it is equally important to develop models that reduce the inference time without any significant drop in performance. To this end, we develop a multi-task learning model for three different entities (*cell-line*, *tissue*, and *strain*) and show that we can reduce the inference time by a factor of 3 without any drop in performance when compared with a single-task model for each entity.

Our main contributions are as follows:

- We fine-tune a multi-task PubMedBERT model, demonstrating a significant reduction in inference time.
- We compare the performance of our multi-task model with that of a single-task model and show that there is no significant drop in F1 scores. Further, we built a multi-class token classification model on our corpus and found that it performs the worst which shows the effectiveness of using a multi-task learning model.
- We release [2] a new gold-standard corpus manually tagged with *cell-line*, *tissue* and *strain* type entity, on which we report our results of the experiments performed. This dataset is the first of its kind that contains manual annotation of *tissue* and *strain* entities.

The rest of the paper is organized as follows. We provide the details of the dataset in section 2. The experiments, results and their analysis are shown in section 3 and 4 respectively. Finally, in section 5, we summarize all the results and provide pointers for future research.

2 Dataset

For the BioNER task, there are several publicly available annotated datasets but the most widely used datasets for benchmarking are JNLPBA (Collier and Kim, 2004; Huang et al., 2020), NCBI-Disease (Dogan et al., 2014), BC5CDR, (Li et al., 2016) BC2GM (Smith et al., 2008), and LINNEAUS (Gerner et al., 2010). These datasets cover mostly *cell line*, *cell type*, *chemical*, *disease*, *gene*, *protein*, and *species* type entities, and most of them rely on PubMed articles as a source. One of the significant concerns regarding most of the BioNER datasets is the data quality, which is not only limited to the biomedical domain. Li et al. (2022) mentioned annotation quality as one of the major challenges in the field of NER. An updated version of the 2004 JNLPBA challenge was released in 2019 to address the flaws in the original corpus (Collier and Kim, 2004; Huang et al., 2020). Another issue is the source and the entity type, which is generally targeted in these datasets. These benchmark datasets lack entities such as *tissue* and *strain* that can help create meaningful cohorts across experiments. This information can be used to control the genetic variability in datasets.

To address the issues mentioned above, we created a gold-standard corpus manually annotated with *cell-line*, *strain*, and *tissue* on abstracts extracted from the Gene Expression Omnibus (GEO) (Edgar et al., 2002) database. It is a public repository established by National Center for Biotech-

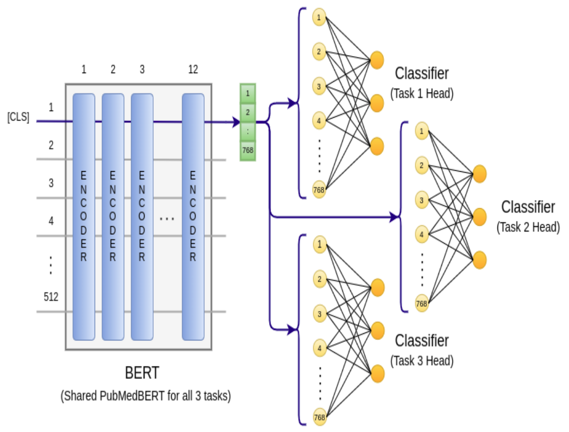


Figure 1: Architecture of the MTL model

Entity type	#Docs	#Words	#Mentions	#Unique Mentions	#Docs w/ at least one mention
strain	3560	234121	3476	574	2049
tissue	2607	430455	1804	338	961
cell-line	3059	532805	1541	483	677

Table 1: Summary statistics of the corpus (includes both the training and the test set.)

[2]The dataset and the source code of our experiments can be found here.

Entity	STL	MTL	Frozen1	Frozen2	Multi-class
Cell line	0.85	0.86	0.70	0.86	0.62
Tissue	0.71	0.71	0.52	0.01	0.46
Strain	0.88	0.87	0.63	0.61	-

Table 2: F1 scores of different models on each entity type.

nology Information (NCBI) for high-throughput gene expression data generated mainly through microarray technology. Several other data applications, such as those that look at genome methylation, chromatin structure, and genome-protein interactions, are now supported by GEO, which has developed along with the quickly changing technological landscape (Clough and Barrett, 2016).

The corpus was manually annotated by the domain experts, and the annotation guidelines followed can be found in Appendix A. The corpus consists of 9226 English paragraphs, and the number of mentions of *strain* (3476) is more than *cell line* (1541) and *tissue* (1804). Despite the less number of total mentions of *cell line*, the number of unique mentions of *cell line* (483) is far greater than the number of unique mentions of *tissue* (338). In the data, wherever the *strain* entity is tagged, the *cell line* and *tissue* are not found, and vice-versa. This is due to the nature of the abstracts (extracted from GEO) where we find either the texts contained mention of *cell line*, *tissue*, *strain* or both *cell line* and *tissue* in the same text. This makes the corpus unique and more reasonable to perform a multi-task learning model instead of building a multi-class token classification model. Table 1 provides more details of the corpus.

3 Experiments

In this section, we describe our experiments in detail about the model architecture, the training procedures, and the evaluation metrics followed.

3.1 MTL Model

Figure 1 shows the MTL architecture deployed in our work. The shared model follows the standard BERT architecture (Devlin et al., 2018) where the task heads consist of two linear layers. The first layer has a shape of 768 x 768, whose outputs are passed through the ReLU activation function and then fed into the second linear layer with a shape of 768 x 3. This layer acts as the token classifier, where each token is assigned one of three classes following the BIO tagging scheme.

3.1.1 Training and Evaluation Metrics

The training and testing split was 70:30. The shared model was initialized with PubMedBERT (Gu et al., 2020) weights, and the task heads were randomly initialized. We then fine-tuned the model for eight epochs at a learning rate of 2e-5 and a batch size of 20. Each batch consisted of examples from the three individual entities mentioned in different paragraphs. Each of the examples in the batch contributed to the loss of the task head for that particular example and to the shared BERT model.

To evaluate the model's performance, we consider each predicted entity as correct only if both the entity boundary and entity types are the same as the ground-truth annotation (i.e., exact match). We then calculate F1 scores for each entity type and report the results.

3.1.2 Controlling other factors

Different factors can affect BERT's inference time, such as batch size, sequence length, choice of deep-learning framework, and hardware. We used a batch size of 1 in all of our experiments, and to control the sequence length, we fixed the corpus that was used to test different model variants, ensuring it resembled production workloads. Regarding hardware, we used an AWS g4dn.xlarge [3] instance as our GPU machine and a laptop with Intel i5-7300U as our CPU machine. For all the experiments, we used Pytorch, [4].

3.2 Single Task Learning (STL) & Multi-class Token Classification Model

To compare the results of our multi-task learning model, we fine-tuned three different individual PubMedBERT models for *cell line*, *tissue*, and *strain* type entities. We refer to these models as single-task learning models as they are fine-tuned for each individual entity.

In general, for NER tasks, a multi-class token classification model is preferable. While in the case of biomedical text, all entities might not be mentioned in the same text; for example, in our case, the corpus did not have *strain* entity wherever there was mention of *tissue* and *cell line* entities. However, since *tissue* and *cell line* annotations were done together, it was possible to compare the results with that of the multi-task model. So, we

[3]https://aws.amazon.com/ec2/instance-types/g4/
[4]https://pytorch.org/

Model	Inference time (CPU)	Inference time (GPU)
Single-task (tissue + cell line + strain)	430 ± 16 ms	31 ± 1 ms
Multi-task	150 ± 6 ms	11 ± 1 ms

Table 3: The inference time per input (avg) of the MTL model compared to the single-task models run sequentially.

fine-tuned a multi-class token classification model combining the *tissue* and *cell line* paragraphs for eight epochs with a learning rate of 2e-5 and batch size of 16.

4 Results and Analysis

The results of our experiments are displayed in Table 2. The single-task learning model (STL) or the PubMedBERT model fine-tuned for three individual entities achieves an F1 score of 0.85, 0.71, and 0.88 for cell line, tissue, and strain, respectively. The third column shows the results of our MTL model fine-tuned jointly on the three tasks, and the F1 scores are 0.86, 0.71, and 0.87 for cell line, tissue, and strain, respectively which shows that there is no significant change in F1 score when compared to the single task model results. The MTL model for the cell line entity gives a better F1 score of 0.86 than the single-task learning model for the cell line. This shows that the MTL model is able to learn the mentions of cell line better than the other entities.

The results of the multi-class token classification model built over the paragraphs containing only cell line and tissue are 0.62 and 0.46, respectively. Since our data is unique in terms of the entities annotated and their mentions in the paragraphs, deploying a multi-class token classification model to learn the properties of the entities in the text is not a good choice in our case as it gives poor results.

It might be possible that the underlying PubMedBERT model learns the same features while fine-tuning for different NER tasks; hence, the MTL model is performing well. To rule out this possibility, we fine-tune the models after freezing the encoder layers of the PubMedBERT model. The fourth and fifth columns of Table 2 show the F1 scores when only the last layer with the task-specific head is trained during the fine-tuning process, and the underlying PubMedBERT layers are frozen. The Frozen1 model is initiated with the pre-trained PubMedBERT weights, and the Frozen2 model is initiated with the model's weights fine-tuned only on the cell line NER task. The F1 scores for the Frozen1 and Frozen2 models are quite poor, which clearly implies that jointly fine-tuning the MTL model on multiple NER tasks learns new features and performs better. The Frozen2 model achieves a good F1 score for the cell line because the underlying frozen model was fine-tuned for the same field.

4.1 3x reduction in inference time

Our primary finding is that an MTL model described above jointly trained on three different NER tasks gives the same model performance when compared to that of a PubMedBERT model fine-tuned separately for three tasks. Table 3 shows the average inference time taken by the MTL model and the single-task model when run sequentially. The MTL model takes around 11 ms on GPU and 150 ms on CPU, which is roughly three times less than the time taken by the single-task model. This shows the primary benefit of joint MTL training, which leads to a considerable reduction in inference time and cost and is crucial for practical applications. Instead of doing a forward pass through 3 separate BERT models to tag a paragraph of text, we only have to do it for one BERT model. The task heads themselves have a negligible contribution to inference time.

4.2 Low prediction accuracy for tissue

As seen in Table 2, the F1 scores for *tissue* field is much lower than that of *cell line* and *strain*. Even the single-task learning model fine-tuned for *tissue* entity gives an F1 score of 0.71 only. There might be two possible reasons for the poor performance. Firstly, *cell line* and *strain* names have a very different sub-word structure as compared to the *tissue* names and thus are significantly easier to detect. Secondly, detecting *tissue* names requires a deeper understanding of the surrounding context in which it occurs. For example, 'blood' can be a tissue, but it can also occur in a different context where it is not a tissue.

In order to see if we can improve the prediction accuracy for the tissue field, we fine-tuned an MTL model with two tasks. One was the actual NER task, and another was an auxiliary classification task that predicted whether an input paragraph had any tissue tag present or not. We tried several combinations of the learning rate, batch size, and weightage of the two tasks in the final loss function, but the best F1 score achieved was still 0.71, as reported in Table 2.

5 Conclusion and Future Work

In this study, we demonstrated how multi-task learning may be used to speed up model inference for complementary tasks that must be performed simultaneously on the same input. In particular, we compared our multi-task model to a single-task model and demonstrated that while the multi-task learning model's performance remained constant, the inference time was reduced by three. Moreover, for our experiments, we created a gold-standard corpus, manually tagged with *cell-line*, *tissue* and *strain*. This corpus is the first of its kind where three different entities are manually curated by domain experts.

When compared to the other entity types, the models' performance in identifying *tissue* names was incredibly poor, demonstrating how challenging it is to extract accurate *tissue* names from the text in the right context. For *tissue* NER, we must either discover a more suitable auxiliary task or develop some rule-based methods that will enhance the entity's overall performance. To increase the accuracy of *tissue*, we intend to carry out these actions in the future. Investigating the MTL model for inference time on benchmark datasets would be another interesting project.

Acknowledgements

We would like to thank the ICON-2022 anonymous reviewers and Shubhra Agrawal for their invaluable comments and feedback on this work. We also thank Amritanjali Kiran at Elucidata for helping in preparing the datasets.

References

Emily Clough and Tanya Barrett. 2016. The gene expression omnibus database. *Methods in molecular biology*, 1418:93–110.

Nigel Collier and Jin-Dong Kim. 2004. Introduction to the bio-entity recognition task at JNLPBA. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP)*, pages 73–78, Geneva, Switzerland. COLING.

Gamal Crichton, Sampo Pyysalo, Billy Chiu, and Anna Korhonen. 2017. A neural network multi-task learning approach to biomedical named entity recognition. *BMC Bioinformatics*, 18.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Rezarta Dogan, Robert Leaman, and Zhiyong lu. 2014. Ncbi disease corpus: A resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47.

Ron Edgar, Michael Domrachev, and Alex E. Lash. 2002. Gene expression omnibus: Ncbi gene expression and hybridization array data repository. *Nucleic acids research*, 30 1:207–10.

Martin Gerner, Goran Nenadic, and Casey Bergman. 2010. Linnaeus: A species name identification system for biomedical literature. *BMC bioinformatics*, 11:85.

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.

Ming-Siang Huang, Po-Ting Lai, Pei-Yen Lin, Yu-Ting You, Richard Tzong-Han Tsai, and Wen-Lian Hsu. 2020. Biomedical named entity recognition and linking datasets: survey and our recent development. *Briefings in Bioinformatics*, 21(6):2219–2238.

Forrest N. Iandola, Albert E. Shaw, Ravi Krishna, and Kurt W. Keutzer. 2020. Squeezebert: What can computer vision teach nlp about efficient neural networks?

Muhammad Raza Khan, Morteza Ziyadi, and Mohamed A. Abdelhady. 2020. Mt-bioner: Multi-task learning for biomedical named entity recognition using deep bidirectional transformers. *ArXiv*, abs/2001.08904.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*.

Jiao Li, Yueping Sun, Robin J. Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J. Mattingly, Thomas C. Wiegers, and Zhiyong Lu. 2016. BioCreative V CDR task corpus: a resource for chemical disease relation extraction. *Database*, 2016. Baw068.

Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2022. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):50–70.

Liyuan Liu, Jingbo Shang, Frank F. Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. In *AAAI Conference on Artificial Intelligence*.

Tahir Mehmood, Alfonso Gerevini, Alberto Lavelli, and Ivan Serina. 2019. *Leveraging Multi-task Learning for Biomedical Named Entity Recognition*, pages 431–444.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

L. Smith, L. Tanabe, R. Ando, C. Kuo, I-Fang Chung, C. Hsu, Y. Lin, Roman Klinger, Christoph Friedrich, K. Ganchev, M. Torii, Hongfang Liu, Barry Haddow, Craig Struble, Richard Povinelli, Andreas Vlachos, William Baumgartner Jr, Lawrence Hunter, B. Carpenter, and W. Wilbur. 2008. Overview of biocreative ii gene mention recognition. *Genome Biology*, 9.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic bert for resource-limited devices.

Xuan Wang, Yu Zhang, Xiang Ren, Yuhao Zhang, Marinka Zitnik, Jingbo Shang, Curtis Langlotz, and Jiawei Han. 2018. Cross-type biomedical named entity recognition with deep multi-task learning. *Bioinformatics*, 35(10):1745–1752.

Mei Zuo and Yang Zhang. 2020. Dataset-aware multi-task learning approaches for biomedical named entity recognition. *Bioinformatics*, 36(15):4331–4338.

Appendix

A Manual Curation Guidelines

For annotating the corpus, two curators were recruited, and both had a biological background. The paragraphs were extracted based on the dataset ids from the GEO database and were exported to the Labelstudio [5] tool for annotations. Each dataset was assigned to the two curators for double-blinded curation where the curators curate the datasets assigned to them independently. The similarities were assessed for every dataset curated by two curators independently and in the case of dissimilarity, the dataset was passed to an expert curator for final annotations. Apart from this, about 10% of datasets were randomly picked for quality checks even if there was no dissimilarity.

The curation for *tissue* and *cell line* was done together and the ontology followed for *tissue* and *cell line* were the BRENDA Tissue Ontology (BTO) [6] and Cellosaurus (CVCL)[7] respectively. In the case of annotating *strain* entity, the strain of mouse and rats used during the experimental process was annotated. To find out the attribute of each mouse and rat provided in the experimental design of the dataset ids, the curators referred to Mouse Genome Informatics (MGI) [8] for the strain information.

B Dataset creation for Multi-class sequence model

Dataset for multi-class token classification model includes paragraphs with tag for *cell-line* (928), *tissue* (1347), *cell-line & tissue* (102), *none* (2557) which was split in 70:30 ratio for training and testing in stratified way.

[5]https://labelstud.io/
[6]https://www.ebi.ac.uk/ols/ontologies/bto
[7]https://www.cellosaurus.org/
[8]http://www.informatics.jax.org/home/strain

English To Indian Sign Language: Rule-Based Translation System Along With Multi-Word Expressions and Synonym Substitution

Abhigyan Ghosh
abhigyan.ghosh@research.iiit.ac.in
Radhika Mamidi
radhika.mamidi@iiit.ac.in

Abstract

The hearing challenged communities all over the world face difficulties to communicate with others. Machine translation has been one of the prominent technologies to facilitate communication with the deaf and hard of hearing community worldwide. We have explored and formulated the fundamental rules of Indian Sign Language(ISL) and implemented them as a translation mechanism of English Text to Indian Sign Language glosses. According to the formulated rules and sub-rules, the source text structure is identified and transferred to the target ISL gloss. This target language is such that it can be easily converted to videos using the Indian Sign Language dictionary. This research work also mentions the intermediate phases of the transfer process and innovations in the process such as Multi-Word Expression detection and synonym substitution to handle the limited vocabulary size of Indian Sign Language while producing semantically accurate translations.

1 Introduction

There are more than 300 sign languages all over the world, depending upon the region of the world (uni). Nearly 4 million deaf people and more than 10 million hard of hearing people in India (Zeshan et al., 2005). Out of such a big number, approximately 1 million deaf adults and around 0.5 million deaf children in India use Indian Sign Language. Rest of nearly 2.5 million deaf and hard of hearing people do not use any sign language to communicate. Thus nearly 80% of hearing-impaired individuals have very limited or no access to education and other information. Hearing-impaired people use sign language using handshapes, fingers, facial expressions, gestures, and other body parts. It is a visual-spatial language as the signer often uses the 3D space around his body to describe an event. Sign languages, until the 1960s, were not viewed as bona fide languages but just collections of gestures and mime. Dr Stokoe's research on American Sign Language proved that it is a full-fledged language with its own grammar, syntax, and other linguistic attributes (Stokoe, 1960). There are some efforts to prove the same for other sign languages, including Indian Sign Language.

The sign language used in India is Indian Sign Language (ISL). A study by (Vasishta et al., 1980) specifies that the ISL used in different parts of India is almost identical in its structure, with differentiation in signs. It is a social need to encourage the hearing impaired individuals of the Indian subcontinent with a tool that can translate the English text to ISL. ISL has all the properties of a natural language and is also considered a natural language like British Sign Language (BSL), American Sign Language (ASL), and Australian Sign Language (AUSLAN). The idea of automatic machine translation in the area of Sign Language translation has been developing very fast in the last two decades as the technology needs to be used for the hearing impaired people all over the world. Most of the researchers experimented with the rule-based machine translation methodology to translate a spoken or written language to Sign Language as this method relies upon the dictionary and the grammar of the source and target languages. Rule-Based Machine Translation is mostly used these days as Indian Sign Language is a low resource communication system and there is a lack of content published in Indian Sign Language. The current state of the art system is described in (Sugandhi et al., 2020) which is also a rule based system. We have tried to improve upon their work.

2 Goal of the Study

The objective is to take a standard English sentence and convert it to text for sign language. The text for sign language represents the given sentence so that anyone can use it to perform the required gestures. This text for sign language can be then used to automate the process of generating sign language videos.

Up till now, most research has focused on the grammar rules. (Agarwal et al., 2015) and (Mishra, 2019) contain extensive lists of grammar transfer rules but this method for a large part has been restricted to very simple sentences. We try to tackle more complicated sentences here which are simple in structure but some of the meanings conveyed by them are complicated and the previously used methods fall short in some regards. We have therefore used approaches such as Multi-Word Expression detection and Synonym substitution to handle hidden meanings within the sentence structure. Not much work has been done in these areas in this field. (Goyal and Goyal, 2016) have used synonyms before but the methods used by them for synonym substitutions are limited by the size of their manually created dictionary of synonyms for certain words. We have used existing English WordNet described by (Miller et al., 1990) for finding synonyms.

3 Methodology

We have broken down the steps into 3 phases: Pre-processing, Grammar Transfer Rules and Post-Processing. We are using Stanza by (Qi et al., 2020), which is a collection of accurate and efficient tools for the linguistic analysis of many human languages. It is created by the Stanford NLP Group. Apart from this, we are also using MWE tokenizer from NLTK by (Bird and Loper, 2004) for MWE tokenization.

3.1 Pre-Processing

In pre-processing step, we take each sentence and pass it through two processing pipelines:

1. First pipeline is the standard Stanza Pipeline described in (Qi et al., 2020) and includes the steps of Tokenization, POS tagging, Dependency Parsing, Named Entity Recognition and Lemmatization.

2. The second pipeline uses the jMWE dataset created by (Kulkarni and Finlayson, 2011) along with the MWETokenizer available as a part of the NLTK project by (Bird and Loper, 2004)

3.2 Grammar Transfer Rules

Here we are trying to convert dependency relations to syntactic relations. The dependency relations in the source language are defined as:
$parent\ \ node(parent\ \ dependency) \rightarrow child\ \ node(child\ \ dependency)$ which are then converted to syntax tree fragments with the notation
$phrase\ tree\ branch \rightarrow (left\ node)(right\ node)$

1. Source: $VERB(root) \rightarrow NOUN(obj)$
   Transformation: $VP \rightarrow [NOUN(obj)][VERB(root)]$
   Example: $He\ \ eats\ \ mangoes \rightarrow HE\ MANGO\ EAT$

   Explanation: Since ISL follows SOV structure, the sentence arrangement should be NP NP VP. That is why we put all the nouns to the left of the verb.

2. Source: $VERB(any) \rightarrow AUX(aux)$
   Transformation: $VP \rightarrow [AUX(aux)][VERB(any)]$
   Example: $He\ \ was\ \ eating \rightarrow HE\ WAS\ EATING$
   All auxiliaries to a verb come before the verb

3. Source: $VERB(any) \rightarrow ADV(advmod)$
   Transformation: $VP \rightarrow [VERB(any)][ADV(advmod)]$
   Example: $He\ \ ran\ \ quickly \rightarrow HE\ RUN\ QUICKLY$
   Explanation: All adverbs to a verb come after the verb

   Note: Rules 2 and 3 would not have appeared in syntactic transformation rules because they do not change the order of the syntax tree. But since we want to extend this to other languages, we need to specify where each child node goes wrt to the head node.

4. Source: $VERB(root) \rightarrow VERB(any)$
   Transformation: $VP \rightarrow [VERB(root)][VERB(any)]$
   This case is written to handle multiple clauses in a sentence. In case of multiple clauses in a sentence, the clauses will be processed in order.

5. Source: $NOUN(any) \rightarrow ADJ(amod)$
   Transformation: $NP \rightarrow [NOUN(any)][ADJ(amod)]$

Example: $He\ has\ a\ blue\ book \rightarrow HE\ BOOK\ BLUE\ HAVE$
Explanation: Adjectives follow the noun they describe

6. Source: $NOUN(any) \rightarrow NUM(nummod)$
Transformation: $NP \rightarrow [NOUN(any)][NUM(nummod)]$
Example: $He\ has\ three\ sons \rightarrow HE\ SON\ THREE\ HAVE$
Numbers are handled in two ways in ISL:
   (a) Reduplication: Repeat the noun to signal plurality
   (b) Numbers follow the noun denoting the quantity

7. Source: $NOUN(any) \rightarrow ANY(acl : relcl)$
Transformation: $NP \rightarrow [NOUN(any)][ANY(acl : relcl)]$
Example: $I\ saw\ the\ book\ which\ you\ bought \rightarrow I\ BOOK\ YOU\ BUY\ SEE$
Explanation: A relative clause is similar to an adjective and hence it comes after a noun.

3.3 Post Processing

Till this stage we have a rough ordering of the signs we need to show. In post processing, we handle special cases, filter out the unwanted words and reduce the words to their root forms.

3.3.1 Interrogative Sentences

Interrogative sentences are signed by first showing the signs of the sentence converted to the imperative form followed by the question word. These question words usually do not have a specific dependency role assigned to them and can occur in many relations depending on the type of sentences. Apart from this, it also becomes very difficult to judge whether or not a wh-word used in a sentence is a question or not, e.g. $I\ know\ what\ he\ is\ talking\ about \rightarrow I\ HE\ TALK\_ABOUT\ KNOW$. In this sentence, the word what is a wh-word but is used as a conjunction. Similarly, questions may sometimes be formed in English without using a wh-word, e.g., $Did\ you\ do\ your\ homework \rightarrow YOU\ HOMEWORK\ DO\ WHAT$. This problem has been extensively discussed by (Aboh et al., 2005) It is difficult to handle such cases, and what we have tried to do is just take the wh-word and pick it and place it at the end of the sentence.

3.3.2 Negative Sentences

Negative sentences are sentences that have a not or no in them. They convey the meaning that the opposite of the event occurs. We apply a similar strategy for negative sentences by just picking and placing the negative word at the end of the sentence, e.g. $He\ is\ not\ a\ doctor \rightarrow HE\ DOCTOR\ NOT$. Similar strategies would work well with other languages too.

As we see, negative sentences and questions are handled similarly in ISL. But then the question arises: If we have a negative question sentence such as $Who\ will\ not\ come\ with\ me$, how is it signed in ISL. We did not find any literature discussing this. So we contacted ISLRTC, and their group of expert signers seemed to sign the negative word before the question word. So the previously mentioned example would be translated as $Who\ will\ not\ go\ with\ me \rightarrow I\ WITH\ GO\ NOT\ WHO$

3.3.3 Synonym Substitution

As we have mentioned earlier, Indian Sign Language is still new, and many terms are yet to be added to the dictionary. As such, till now, there are only 10000 words in the ISL dictionary. As a comparison, English has around 200,000 words (Brewer, 1993) which is 20 times the size. Hence, there are many words in English that do not have an equivalent sign. If such a word occurs in a sentence, it needs to be translated somehow. For that, we substitute those words that are not in the video dictionary with words close to the original word's meaning and have a corresponding word in the dictionary. Closest words are selected from the synset in the WordNet based on a score denoting how similar two word senses are, based on the shortest path that connects the senses in the is-a (hypernym/hypnoym) taxonomy.

3.3.4 Stopword Removal and Lemmatization

After all the above processing is done, we need to clean up the sentences. ISL does not contain words such as articles and certain functional words that do not necessarily convey meaning. But not all stop-words in English are removed from ISL. For this purpose, we created our own list of stop-words for English based on inputs from translations of sentences in our dataset by expert ISL signers. Followed by the removal of the stop-words, we convert all the remaining tokens into their root forms. At the end of this step we reach the point where we are ready with a textual representation of the videos required to be shown for translating the sentence.

3.3.5 Video Translation

After the sentences have been converted into text for sign language, the output tokens are then searched for in the video database. Two cases can arise here; one is that the tokens are available in the database. If the tokens are not available in the database, then we use WordNets to check if there are synonyms for that given word that have the same POS and are present in the dictionary. We assume that if the word has the same POS and is of similar meaning, it can be directly replaced by the synonym word without affecting the grammar. This is corroborated with evidence from the hearing impaired community. If any of the words' synonyms are not found in the dictionary, then that sentence cannot be translated into ISL, unfortunately, and there is little we can do about it. One possibility is to fingerspell the word, but that approach does not give accurate results always. We then concatenate the video for each sign together to generate the final output.

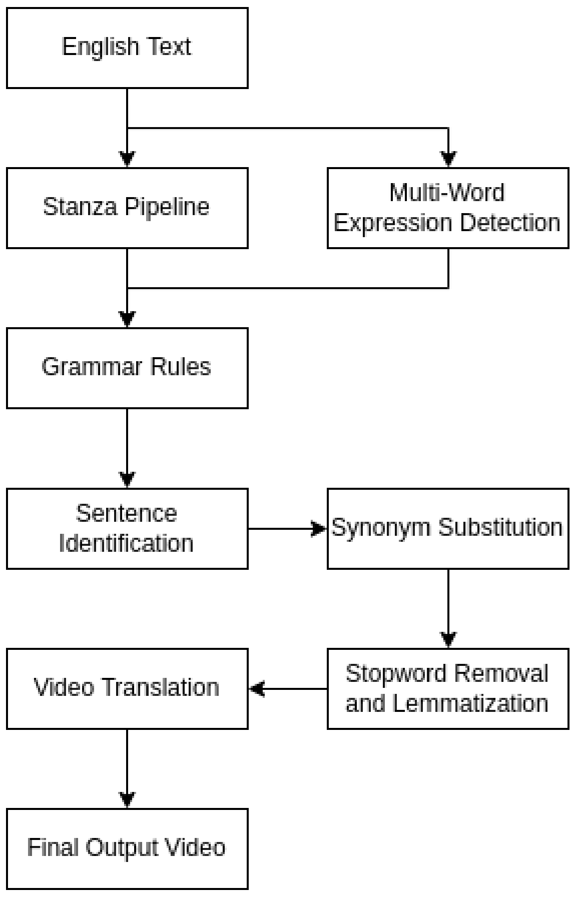


Figure 1: End-to-end System Diagram

		Accuracy
Total Sentences Checked	741	
Sentences with synonyms substituted	304	
Sentences with MWEs detected	24	95.8333

Table 1: Summary of results

4 Results and Discussion

As there was no previously established standard for this task, we selected the simple-wiki-dataset. We ran our model on 741 sentences with lengths of about 5-9 words per sentence. The results were manually evaluated.

4.1 Synonym Substitution

1368 words were substituted by synonyms in 1011 sentences. Some of the substitutions were very accurate. Such as raise→lift, argument→debate, survive→last, astound→amaze, etc. However, not all substitutions were meaning conserving eg. sword→steel, offend→break. Sometimes the substitutions were wrong because the meaning of the sign and the word were totally different. For example in some sentences, *say* became *'state'* but the sign for *'state'* is for the noun *state(country)* and not the verb *state* which actually would have shared the same meaning. Some examples are shown in Table 2

4.2 MWE Processing

There were 426 words in which were identified as Multi-Word Expressions. All of the MWEs identified were correct. Example: $look\_up\_to$, $rolling\_stock$, you_know, $thank\_you$, etc.

5 Conclusion and Future Work

In this paper we have tried to tackle semantically complex sentences and we see that the previously used methods fall short in some regards. Therefore, we have used approaches such as Multi-Word Expression detection and Synonym substitution to handle hidden meanings within the sentence structure. We needed simple English sentences to test out our algorithm, as the rules worked best for simple sentences. In future, we intend to use machine learning approaches to do the translations on a large dataset to convert the standard English sentences. The methods mentioned in the paper could be a way to convert a large dataset of sentences on which Deep Learning models such as sequence2sequence and transformers can be trained.

Sentence	After Grammar Rules	Missing Words	Replacement Word	Final Output
A woman sells newspapers	Woman newspaper sells	newspaper	paper	woman paper sell
Helene let her fall	Helene her let fall	let	allow	H E L E N E mine allow fall
This foreshadows later events	This events later foreshadows	foreshadow	predict	this event late predict

Table 2: Result of Synonym Substitution

References

International day of sign languages.

Enoch Aboh, Roland Pfau, Ulrike Zeshan, et al. 2005. When a wh-word is not a wh-word: The case of indian sign language. *The yearbook of South Asian languages and linguistics*, 2005:11–43.

Sumeet R Agarwal, Sagarkumar B Agrawal, and Akhtar M Latif. 2015. Sentence formation in nlp engine on the basis of indian sign language using hand gestures. *International Journal of Computer Applications*, 116(17).

Steven Bird and Edward Loper. 2004. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain. Association for Computational Linguistics.

Charlotte Brewer. 1993. The second edition of the oxford english dictionary. *The Review of English Studies*, 44(175):313–342.

Lalit Goyal and Vishal Goyal. 2016. Automatic translation of English text to Indian Sign Language synthetic animations. In *Proceedings of the 13th International Conference on Natural Language Processing*, pages 144–153, Varanasi, India. NLP Association of India.

Nidhi Kulkarni and Mark Finlayson. 2011. jMWE: A Java toolkit for detecting multi-word expressions. In *Proceedings of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World*, pages 122–124, Portland, Oregon, USA. Association for Computational Linguistics.

George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to wordnet: An on-line lexical database. *International journal of lexicography*, 3(4):235–244.

Gouri Sankar Mishra. 2019. English text to indian sign language translation system. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, 8(10S):460–467.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages.

William C. Jr. Stokoe. 1960. Sign language structure, an outline of the visual communications systems of american deaf. *Studies in Linguistics Occasional paper*, 8.

Sugandhi, Parteek Kumar, and Sanmeet Kaur. 2020. Sign language generation system based on indian sign language grammar. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 19(4).

Madan Vasishta, James Woodward, and Susan De Santis. 1980. *An introduction to Indian sign language:(Focus on Delhi)*. All India Federation of the Deaf.

Ulrike Zeshan, Madan N Vasishta, and Meher Sethna. 2005. Implementation of indian sign language in educational settings. *Asia Pacific Disability Rehabilitation Journal*, 16(1):16–40.

Improving Contextualized Topic Models with Negative Sampling

Suman Adhya, Avishek Lahiri, Debarshi Kumar Sanyal
Indian Association for the Cultivation of Science, Jadavpur, Kolkata-700032, India
{ adhyasuman30, avisheklahiri2014, debarshisanyal }@gmail.com

Partha Pratim Das
Ashoka University, Sonipat, Haryana-131029, India
Indian Institute of Technology Kharagpur, West Bengal-721302, India
ppd@cse.iitkgp.ac.in

Abstract

Topic modeling has emerged as a dominant method for exploring large document collections. Recent approaches to topic modeling use large contextualized language models and variational autoencoders. In this paper, we propose a negative sampling mechanism for a contextualized topic model to improve the quality of the generated topics. In particular, during model training, we perturb the generated document-topic vector and use a triplet loss to encourage the document reconstructed from the correct document-topic vector to be similar to the input document and dissimilar to the document reconstructed from the perturbed vector. Experiments for different topic counts on three publicly available benchmark datasets show that in most cases, our approach leads to an increase in topic coherence over that of the baselines. Our model also achieves very high topic diversity.

1 Introduction

The modern world is witnessing tremendous growth in digital documents. It is often necessary to organize them into semantic categories to make the content more easily accessible to users. The assignment of domain tags through manual intervention can be quite cumbersome and very expensive to maintain, mainly due to the enormity and diversity of the available data. The use of topic modelling techniques can be of huge significance in this area because of their ability to automatically learn the overarching themes or topics from a collection of documents in an unsupervised way and tag the documents with their dominant topics (Newman et al., 2010; Boyd-Graber et al., 2017; Adhya and Sanyal, 2022). Informally, a topic is a group of extremely related words. While latent Dirichlet allocation (LDA) (Blei et al., 2003) is the classical topic modeling approach, recently neural topic models have become popular as they decouple the inference mechanism from the underlying modeling assumptions (e.g., the topic prior), thereby simplifying the design of new topic models. Neural topic models are based on variational autoencoders (VAEs) (Kingma and Welling, 2014) and allow us to leverage the progress in deep learning in modeling text (Zhao et al., 2021). The recently proposed contextualized topic model (CTM) (Bianchi et al., 2021), which is a neural topic model, represents each document in the collection both as a bag-of-words (BoW) vector as well as a dense vector produced by a pre-trained transformer like sentence-BERT (SBERT) (Reimers and Gurevych, 2019), thus combining a classical representation with a contextualized representation that captures the semantics of the text better. CTM produces state-of-the-art performance on many benchmark datasets (Bianchi et al., 2021).

A neural topic model is trained to maximize the log-likelihood of the reconstruction of the input document and minimize the KL-divergence of the learned distribution of the latent (topic) space from a known prior distribution of the latent space. If the topics in a document are perturbed, that is, say, the top topic in a document is deleted, the document should display a marked change in its word distribution. Such an objective is not explicitly modeled above. In this paper, we train CTM to infer topics from a document in such a way that while the inferred topics should aid in reconstructing the document (as in any topic modeling algorithm), when the top topics are perturbed it should fail to reconstruct the original document. This is done by treating the document reconstructed from the correct topic vector as an anchor that is encouraged to be similar to the original input document but dissimilar to the document reconstructed from the perturbed topics. Our proposed model, **CTM-Neg**, achieves higher average topic coherence, measured by NPMI score, than that of other competing topic models, and very high topic diversity on three datasets. We have made our code publicly available[1].

Thus, our primary contributions are:

1. We propose a *simple but effective negative sampling technique for neural topic models*. Negative samples are produced automatically in an unsupervised way.

2. We perform extensive experiments on three publicly available datasets. In particular, we compare the proposed model with four other topic models for eight different topic counts on each dataset. We observe that the proposed strategy *leads to an increase in topic coherence* over the baselines in most of the cases. Averaged over different topic counts, CTM-Neg achieves the highest mean NPMI score on all three datasets, and highest mean CV on two datasets, and the second highest mean CV on the third. CTM-Neg also attains the best or the second best mean topic diversity scores on the three datasets though all the topic models except one (which underperforms) produce similar high topic diversity.

2 Related Work

Latent Dirichlet allocation (LDA) (Blei et al., 2003) models every document in a given corpus as a mixture of topics, where each topic is a probability distribution over the vocabulary. Among the modern neural alternatives to LDA, a pioneering approach is the ProdLDA model (Srivastava and Sutton, 2017). It is a VAE-based topic model that uses an approximate Dirichlet prior (more precisely, Laplace approximation to the Dirichlet prior in the softmax basis), instead of a standard Gaussian prior (Miao et al., 2016). The VAE takes a bag-of-words (BoW) representation of a document, maps it to a latent vector using an encoder or inference network, and then maps the vector back to a discrete distribution over words using a decoder or generator network. CTM (Bianchi et al., 2021) augments ProdLDA by allowing in its input a contextualized representation (SBERT) of the documents. Embedded topic model (ETM) (Dieng et al., 2020) is a VAE-based topic model that uses distributed representations of both words and topics.

Negative sampling in NLP-based tasks was popularized after its use in the word embedding model, word2vec (Mikolov et al., 2013). The idea of negative sampling is to 'sample' examples from a noise distribution and ensure that the model being trained can distinguish between the positive and negative examples. It can be used to reduce the computational cost of training, help identify out-of-distribution examples, or to make the model more robust to adversarial attacks (Xu et al., 2022). A few works have recently applied it to topic modeling. For example, (Wu et al., 2020) proposed a negative sampling and quantization model (NQTM) with a modified cross-entropy loss to generate sharper topic distributions from short texts. Some researchers have applied generative adversarial networks to design topic models (Wang et al., 2019; Hu et al., 2020; Wang et al., 2020), but since the negative examples are generated from an assumed fake distribution, they bear little similarity to real documents. In (Nguyen and Luu, 2021), a negative document sample is created by replacing the weights of the words having the highest tf-idf scores in the input document with the weights of the same words in the reconstructed document. Our method follows a different strategy: it generates a perturbed document-topic vector (instead of an explicit negative document) and uses triplet loss to push the BoW vector reconstructed from the correct topic vector closer to the input BoW vector and farther from the BoW vector generated from the perturbed topics. Unlike the present work, none of the other adversarial topic models use contextual embeddings as input.

3 Proposed Method

3.1 Baseline Architecture

Our proposed model is based on a VAE architecture. In particular, we build upon CTM (Bianchi et al., 2021). We assume that the vocabulary size is V and a document is represented as a normalized bag-of-words vector $\mathbf{x}_{\mathrm{BoW}}$ as well as a contextualized embedding vector $\mathbf{x}_c$. A linear layer converts $\mathbf{x}_c$ to a V-dimensional vector. The encoder of the VAE concatenates these two vectors into a single $2V$-dimensional vector $\mathbf{x}$ and outputs the parameters of the posterior $\left(\boldsymbol{\mu}_{T\times 1}, \boldsymbol{\Sigma}_{T\times 1}\right)$ where T is the number of topics, $\boldsymbol{\mu}_{T\times 1}$ denotes the mean, and $\boldsymbol{\Sigma}_{T\times 1}$ represents the diagonal covariance matrix. Note that it is standard in the VAE literature to assume a diagonal covariance matrix instead of a full covariance matrix (Srivastava and Sutton, 2017). In the decoder, using the reparameterization trick

[1] `https://github.com/AdhyaSuman/CTMNeg`

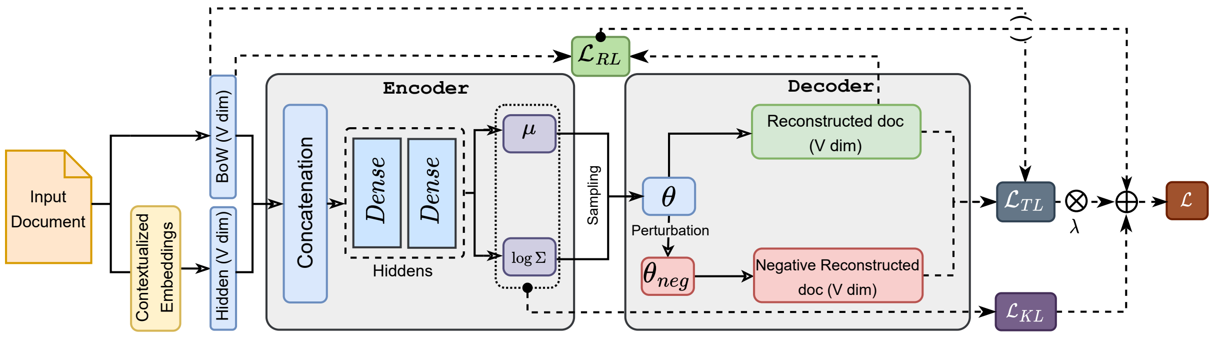


Figure 1: Framework for the contextualized topic model with negative sampling (CTM-Neg).

the latent representation $(\mathbf{z}_{T\times 1})$ is generated:

$$\mathbf{z}_{T\times 1} = \boldsymbol{\mu}_{T\times 1} + \boldsymbol{\Sigma}^{1/2}_{T\times 1} \odot \boldsymbol{\epsilon}_{T\times 1} \qquad (1)$$

where $\boldsymbol{\epsilon}_{T\times 1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\odot$ denotes Hadamard product. This hidden representation $\mathbf{z}$ is then used as a logit of a softmax function $(\sigma(\cdot))$ to generate the document-topic distribution $\boldsymbol{\theta}_{T\times 1}$ $(= \sigma(\mathbf{z}_{T\times 1}))$. The decoder has an unnormalized topic-word matrix $\boldsymbol{\beta}_{T\times V}$, which is used to reconstruct the word distribution in the following manner:

$$\hat{\mathbf{x}}_{V\times 1} = \sigma(\boldsymbol{\beta}^{\top}_{T\times V}\boldsymbol{\theta}_{T\times 1}) \qquad (2)$$

To formulate the loss function, note that the encoder learns the posterior distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. We assume that the prior is $p(\mathbf{z})$. The decoder is the generative model $p_{\boldsymbol{\theta}}(\mathbf{x}_{\text{BoW}}|\mathbf{z})$. The loss function to be minimized is given by

$$\begin{aligned}\mathcal{L}_{\text{CTM}} &= \mathcal{L}_{\text{RL}} + \mathcal{L}_{\text{KL}} \\ &\equiv -\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} p_{\boldsymbol{\theta}}(\mathbf{x}_{\text{BoW}}|\mathbf{z}) \\ &\quad + \text{D}_{\text{KL}}\left(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right) \qquad (3)\end{aligned}$$

here, the first term $(\mathcal{L}_{\text{RL}})$ is the reconstruction loss (measured by the cross-entropy between the predicted output distribution $\hat{\mathbf{x}}$ and the input vector $\mathbf{x}_{\text{BoW}}$) while the second term $\mathcal{L}_{\text{KL}}$ is the KL-divergence of the learned latent space distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ from the prior $p(\mathbf{z})$ of the latent space.

3.2 Proposed Negative Sampling Mechanism

To improve the topic quality, we train the above model with negative samples as follows. For every input document, after a topic vector $\boldsymbol{\theta}$ is sampled, a perturbed vector $\tilde{\boldsymbol{\theta}}_{\text{neg}}$ is generated from it by setting the entries for the top S topics (i.e., the S positions in $\boldsymbol{\theta}$ corresponding to the S largest values in $\boldsymbol{\theta}$) to zero. $\tilde{\boldsymbol{\theta}}_{\text{neg}}$ is then normalized so that the resulting vector $\boldsymbol{\theta}_{\text{neg}}$ is a probability vector. The normalization is done simply by dividing the values in $\tilde{\boldsymbol{\theta}}_{\text{neg}}$ by their sum, as all values, in $\tilde{\boldsymbol{\theta}}_{\text{neg}}$ are already non-negative (since $\boldsymbol{\theta}$ is obtained by applying `softmax`). Mathematically,

$$\boldsymbol{\theta}_{\text{neg}} = \frac{\tilde{\boldsymbol{\theta}}_{\text{neg}}}{\sum_{i=1}^{T}\tilde{\theta}_{\text{neg}}[i]} \qquad (4)$$

$$\text{where, } \tilde{\theta}_{\text{neg}}[i] = \begin{cases} 0 & \text{if } i \in \text{argmax}(\boldsymbol{\theta}, S) \\ \theta[i] & \text{otherwise} \end{cases}$$

The function `argmax`$(\boldsymbol{\theta}, S)$ returns the indices of the S largest values in $\boldsymbol{\theta}$. We treat S as a hyperparameter. Like $\boldsymbol{\theta}$, the perturbed topic vector $\boldsymbol{\theta}_{\text{neg}}$ is passed through the decoder network. The latter generates $\hat{\mathbf{x}}_{\text{neg}} = \sigma(\boldsymbol{\beta}^{\top}\boldsymbol{\theta}_{\text{neg}})$. We introduce a new term, triplet loss $\mathcal{L}_{\text{TL}}$, in Eq. (3) assuming the anchor is $\hat{\mathbf{x}}$, the positive sample is $\mathbf{x}_{\text{BoW}}$ (the original input document), and the negative sample is $\hat{\mathbf{x}}_{\text{neg}}$:

$$\mathcal{L}_{\text{TL}} = \max(||\hat{\mathbf{x}} - \mathbf{x}_{\text{BoW}}||_2 - ||\hat{\mathbf{x}} - \hat{\mathbf{x}}_{\text{neg}}||_2 + m, 0) \qquad (5)$$

where m is the margin. Therefore, the modified loss function to be minimized is given by:

$$\mathcal{L} = (\mathcal{L}_{\text{RL}} + \mathcal{L}_{\text{KL}}) + \lambda\mathcal{L}_{\text{TL}} \qquad (6)$$

where λ is a hyperparameter. Fig. 1 depicts the proposed model. The model is trained in an end-to-end manner using Adam optimizer and backpropagation.

4 Experimental Setup

We perform all experiments in OCTIS (Terragni et al., 2021), which is an integrated framework for topic modeling.

4.1 Datasets

We use the following three datasets:

Dataset	#Topics							
	10	20	30	40	50	60	90	120
GN	(2, 0.7)	(2, 0.58)	(2, 0.59)	(2, 0.59)	(3, 0.82)	(3, 0.94)	(1, 0.68)	(3, 0.82)
20NG	(3, 0.78)	(3, 0.83)	(3, 0.86)	(1, 0.74)	(1, 0.12)	(3, 0.27)	(1, 0.84)	(1, 0.90)
M10	(3, 0.9)	(3, 0.49)	(1, 0.82)	(1, 0.59)	(3, 0.82)	(3, 0.58)	(3, 0.93)	(3, 0.27)

Table 1: Each paired entry shows the best hyperparameters (S, λ) in CTM-Neg as discovered by OCTIS for a given (Dataset, #Topics) combination.

1. **GoogleNews** (**GN**): It consists of $11,109$ news articles, titles, and snippets collected from the Google News website in November 2013 (Qiang et al., 2020).

2. **20NewsGroups** (**20NG**): It comprises $16,309$ newsgroup documents partitioned (nearly) evenly across 20 different newsgroups (Terragni et al., 2021).

3. **M10**: It is a subset of CiteSeer^X data comprising 8355 scientific publications from 10 distinct research areas (Pan et al., 2016).

The last two datasets are available in OCTIS while we added the first one.

4.2 Evaluation Metrics

Coherence measures help to assess the relatedness between the top words of a topic. Informally, a topic is said to be coherent if it contains words that, when viewed together, help humans to recognize it as a distinct category (Hoyle et al., 2021). We use **Normalized Pointwise Mutual Information** (**NPMI**) and (Lau et al., 2014) and **Coherence Value** (**CV**) (Röder et al., 2015) to measure topic coherence. NPMI is widely adopted as a proxy for human judgement of topic coherence though some researchers also use CV (but CV has some known issues). NPMI calculates topic coherence by measuring how likely the topic words are to co-occur. If $p(w_i, w_j)$ represents the probability of two words w_i and w_j co-occurring in a boolean sliding context window, and $p(w_i)$ is the marginal probability of word w_i, then the NPMI score is given by (Lau et al., 2014),

$$\text{NPMI}(w_i, w_j) = \left(\frac{\log \frac{p(w_i,w_j)+\epsilon}{p(w_i).p(w_j)}}{-\log(p(w_i, w_j) + \epsilon)} \right) \quad (7)$$

where ϵ is a small positive constant used to avoid zero. $\text{NPMI}(w_i, w_j)$ lies in $[-1, \ +1]$ where -1 indicates the words never co-occur and $+1$ indicates they always co-occur. CV is calculated using an indirect cosine measure along with the NPMI score over a boolean sliding window (Röder et al., 2015; Krasnashchok and Jouili, 2018). OCTIS uses the `CoherenceModel` of gensim where NPMI is referred to as `c_npmi` and CV as `c_v`.

We measure the diversity of topics using **Inversed Rank-Biased Overlap** (**IRBO**) (Bianchi et al., 2021). It gives 0 for identical topics and 1 for completely dissimilar topics. Suppose we are given a collection $\aleph$ of T topics where each topic is a list of words such that the words at the beginning of the list have a higher probability of occurrence (i.e., are more important or more highly ranked) in the topic. Then, the IRBO score of the topics is defined as

$$\text{IRBO}(\aleph) = 1 - \frac{\sum_{i=2}^{T} \sum_{j=1}^{i-1} \text{RBO}(l_i, l_j)}{n} \quad (8)$$

where $n = \binom{T}{2}$ is the number of pairs of lists, and $\text{RBO}(l_i, l_j)$ denotes the standard Rank-Biased Overlap between two ranked lists l_i and l_j (Webber et al., 2010). IRBO allows the comparison of lists that may not contain the same items, and in particular, may not cover all items in the domain. Two lists (topics) with overlapping words receive a smaller IRBO score when the overlap occurs at the highest ranks of the lists than when they occur at lower ranks. IRBO is implemented in OCTIS. Higher values of NPMI, CV, and IRBO are better than lower values.

In our experiments, for evaluation using the above metrics in OCTIS, we use the top-10 words from every topic and the default values for all the other parameters.

4.3 Baselines and Configuration

We denote our proposed topic model by **CTM-Neg**. As baselines we use the following topic models, which are already implemented in OCTIS:

1. **CTM** (Bianchi et al., 2021).

2. **ProdLDA** (Srivastava and Sutton, 2017).

3. **ETM** (Dieng et al., 2020).

4. **LDA** (Blei et al., 2003).

In CTM-Neg, CTM, and Prod-LDA, the encoder is a fully-connected feedforward neural network (FFNN) with two hidden layers with 100 neurons each, and the decoder is a single-layer FFNN. We use `paraphrase-distilroberta-base-v2` (which is an SBERT model) to obtain the contextualized representations of the input documents in CTM and CTM-Neg.

In CTM-Neg, we set $m = 1$ in Eq. (5) as is the default in PyTorch. We have optimized the hyperparameters S and λ using the Bayesian optimization framework of OCTIS to maximize NPMI. Table 1 shows the optimal values discovered when $S \in \{1, 2, 3\}$ and $\lambda \in [0, 1]$. In LDA, we use 5 passes over the input corpus as the default single pass produces too poor topics. Other hyperparameters are set to their default values in OCTIS. For all datasets, the vocabulary is set to the most common 2K words in the corpus. Experiments for each topic model are done for all topic counts in the set $\{10, 20, 30, 40, 50, 60, 90, 120\}$. We have trained all models for 50 epochs.

5 Results

5.1 Quantitative Evaluation

Given a dataset and a topic model, we recorded the median values of NPMI, CV, and IRBO over 5 independent runs for each topic count. We choose median instead of mean as the former is more robust to outliers. Then for the same dataset and topic model, we compute the average of these values so that we can get an idea of the performance of the topic model without coupling it to a specific topic count. Table 2 shows the corresponding values where we mention the median along with the mean.

We observe that CTM-Neg achieves the highest average NPMI on all datasets. CTM-Neg also produces the highest average CV on all datasets except M10 where CTM performs slightly better. In the case of IRBO, while CTM-Neg gives the highest scores on GN and 20NG, it ranks as the second best on M10. It is also observed that the IRBO values for all models except ETM are very close to each other.

In order to afford a more fine-grained view of the performance of the topic models, Fig. 2 depicts how the scores vary with topic count for all topic models and on all datasets. CTM-Neg always achieves the highest NPMI and CV scores on GN and 20NG datasets. On the M10 corpus, CTM scores slightly better than CTM-Neg in NPMI and CV for some topic counts. The IRBO plots in Fig. 2 show that on a given dataset, all topic models, except ETM, achieve very similar IRBO scores for every topic count. ETM is always found to produce significantly lower IRBO values. CTM-Neg does not always produce the highest IRBO. For example, on the M10 corpus, the IRBO score of CTM-Neg is the highest till $T = 20$ after which LDA dominates and CTM-Neg is relegated to the second position. A closer look at Fig. 2 reveals that this gain in topic diversity for LDA comes at the expense of reduced NPMI.

5.2 Extrinsic Evaluation

We also use an extrinsic task to evaluate the topic models. We measure the predictive performance of the generated topics on a document classification task. Specifically, we use the M10 dataset from OCTIS where each document is already marked with one of 10 class labels as shown in Table 3. The corpus is divided into train/dev/test subsets in the ratio 70:15:15. Each topic model is trained on the training subset to produce $T = 10$ topics and the T-dimensional document-topic latent vector is used as a representation of the document. Next, a linear support vector machine is trained with these representations of the training subset (for each topic model), and the performance on the test subset is recorded. Fig. 3 shows that CTM-Neg achieves the highest accuracy.

5.3 Qualitative Evaluation

It is acknowledged in the NLP community that automatic metrics do not always accurately capture the quality of topics produced by neural models (Hoyle et al., 2021). So we perform manual evaluation of the topics for a few selected cases. Table 4 shows some of the topics output by random runs of the different topic models on 20NG for $T = 20$ topics. Note that the table displays manually aligned topics, that is, the first topic mentioned against any of the topic models is very similar to the first topic stated against every other topic model, and similarly for all other topics. We observe that the topics generated by CTM-Neg contain very specific words in the top positions that distinguish the topics more clearly compared to the case of other models.

Dataset	Model	Coherence				Diversity	
		NPMI		CV		IRBO	
		Mean	Median	Mean	Median	Mean	Median
GN	CTM-Neg	**0.142**	**0.188**	**0.530**	**0.552**	**0.998**	**0.998**
	CTM	0.081	0.128	0.485	0.513	0.995	0.995
	ProdLDA	0.056	0.076	0.471	0.476	0.996	0.996
	ETM	-0.263	-0.271	0.414	0.416	0.627	0.660
	LDA	-0.164	-0.176	0.403	0.405	0.997	**0.998**
20NG	CTM-Neg	**0.121**	**0.127**	**0.648**	**0.653**	**0.991**	**0.991**
	CTM	0.093	0.098	0.627	0.632	0.990	0.990
	ProdLDA	0.080	0.084	0.609	0.607	0.990	**0.991**
	ETM	0.049	0.048	0.528	0.527	0.819	0.808
	LDA	0.075	0.080	0.571	0.577	0.983	0.990
M10	CTM-Neg	**0.052**	**0.056**	0.462	**0.461**	0.986	0.985
	CTM	0.048	0.047	**0.466**	**0.461**	0.980	0.979
	ProdLDA	0.025	0.023	0.448	0.449	0.983	0.981
	ETM	-0.056	-0.062	0.345	0.350	0.502	0.484
	LDA	-0.192	-0.201	0.386	0.389	**0.989**	**0.992**

Table 2: Comparison of topic models on three datasets. For each metric and each topic model, we mention the mean and the median of the scores for topic counts {10, 20, 30, 40, 50, 60, 90, 120}.

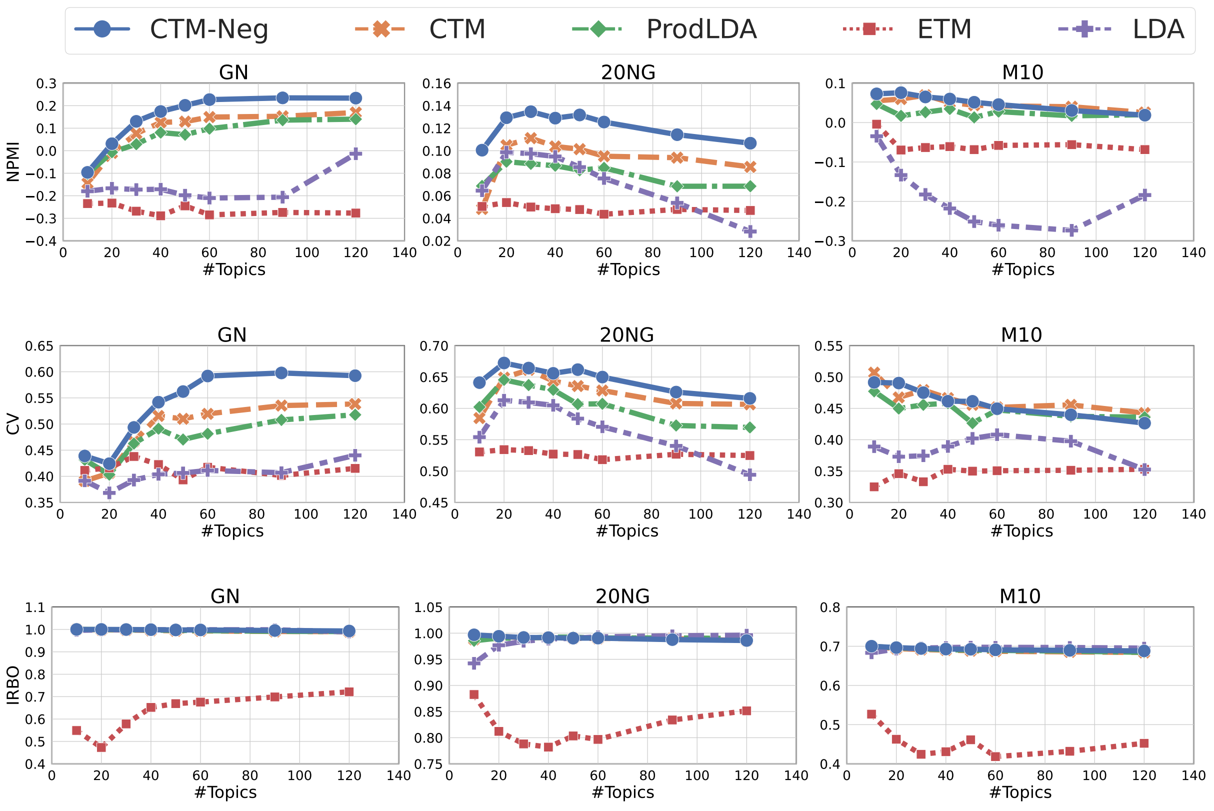


Figure 2: Variation of topic coherence (NPMI and CV) and topic diversity (IRBO) with topic count for different topic models on three datasets. The ordinate value of each data point reports the median over five independent runs.

Label	#Documents
Agriculture	643
Archaeology	131
Biology	1059
Computer Science	1127
Financial Economics	978
Industrial Engineering	944
Material Science	873
Petroleum Chemistry	886
Physics	717
Social Science	997

Table 3: M10 labels with corresponding document counts.

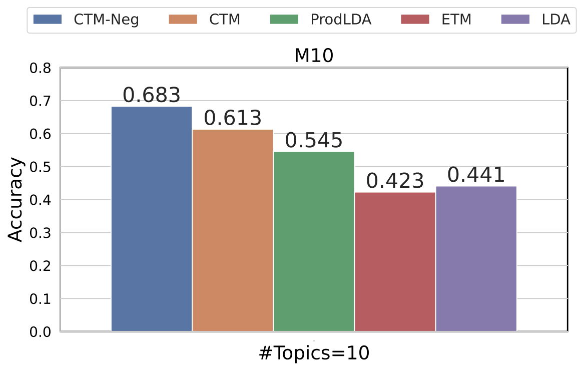


Figure 3: Document classification for M10 corpus with $T = 10$ topics.

For example, the first topic produced by CTM-Neg contains very focused terms like ‘turkish’, ‘israeli’, ‘genocide’, ‘war’, etc., and is easily identifiable as ‘middle-east conflict’ (corresponds to newsgroup `talk.politics.mideast` of 20NG corpus). CTM outputs a very similar topic but it seems to focus only on the ‘Armenian genocide’ yet contains more generic terms like ‘neighbor’ and ‘town’. ProdLDA also focuses primarily on ‘Armenian genocide’ but its last word ‘jewish’ probably refers to the Israeli conflict. While the corresponding topic from LDA contains some generic terms like ‘man’, ‘kill’, etc., most of the words in ETM like ‘kill’, ‘gun’, and ‘fire’ are very general. Moreover, words like ‘leave’ and ‘start’ that occur in this topic in ETM reduce the interpretability of the topic.

Similarly, the fourth topic in CTM-Neg is sports-related and contains specific words like ‘hockey’ and ‘baseball’. While the corresponding topic from ProdLDA mentions ‘hockey’ (but not ‘baseball’), none of the other models produces these terms. The ability of CTM-Neg to extract focused words is probably a consequence of the negative sampling algorithm that encourages a topic to capture the most salient words of its representative documents so that deleting the topic pushes the reconstructed document away from the input document.

Table 5 shows the topics that are discovered in a random run of each topic model on the M10 dataset for $T = 10$ topics. We show four topics – the first is on ‘neural and evolutionary computing’ (or ‘artificial intelligence’), the second on ‘microarray gene expression’, the third on ‘stock market’, and the fourth on ‘multi-agent decision making’. The topics generated by CTM and CTM-Neg are very similar. However, the presence of words like ‘processing’ in the first topic, ‘work’ in the third topic, and ‘approach’ in the fourth topic in CTM appear less connected to the other words in the respective topics. Such outliers are not visible in the topics produced by CTM-Neg. Moreover, the second topic output by CTM-Neg contains very domain-specific terms like ‘dna’ and ‘motif’, which are not produced by CTM. Similar issues occur in ProdLDA and LDA. In the case of ETM, the first topic contains words that make it a mixture of the first two topics produced by the other models. For example, it contains words like ‘neural’ and ‘network’ that occur in the first topic in the other models, and also ‘gene’ and ‘expression’ which are present in the second topic in the other models.

Model	Topics
CTM-Neg	turkish, armenian, jewish, population, muslim, village, israeli, genocide, government, war chip, key, encryption, government, clipper, phone, security, privacy, escrow, secure video, monitor, vga, port, modem, apple, driver, card, resolution, board score, playoff, period, play, fan, win, hockey, game, baseball, lose
CTM	people, armenian, soldier, village, turkish, massacre, troop, neighbor, town, genocide chip, clipper, encryption, government, encrypt, algorithm, agency, secure, phone, key draw, mouse, advance, convert, font, screen, button, host, code, terminal game, win, final, goal, period, cap, score, fan, lead, play
ProdLDA	genocide, armenian, turkish, greek, muslim, village, population, russian, massacre, jewish encryption, secret, secure, chip, privacy, government, key, agency, security, encrypt monitor, card, apple, video, sale, price, board, audio, offer, external game, team, division, season, hockey, playoff, score, goal, player, wing
ETM	people, kill, child, gun, armenian, fire, man, time, leave, start key, chip, encryption, clipper, bit, government, algorithm, message, law, system drive, card, disk, system, bit, run, window, scsi, driver, monitor game, play, win, team, player, year, good, score, hit, season
LDA	people, jewish, armenian, child, man, kill, woman, death, turkish, israeli key, chip, encryption, government, security, clipper, bit, public, message, system card, work, monitor, system, driver, problem, run, machine, video, memory game, team, play, player, win, year, good, season, hit, score

Table 4: Some related topics discovered by different topic models in the 20NG corpus when run for $T = 20$ topics.

Model	Topics
CTM-Neg	neural, network, learn, recurrent, learning, artificial, language, evolutionary, genetic, adaptive expression, gene, datum, sequence, cluster, protein, microarray, dna, analysis, motif stock, return, market, price, volatility, exchange, rate, interest, option, monetary decision, make, agent, making, group, multi, uncertainty, robot, intelligent, autonomous
CTM	network, neural, learn, learning, artificial, evolutionary, language, recurrent, knowledge, processing gene, expression, datum, model, analysis, microarray, cluster, clustering, genetic, classification market, stock, price, return, risk, financial, rate, option, work, volatility decision, agent, make, making, multi, human, group, uncertainty, social, approach
ProdLDA	network, neural, learn, recurrent, artificial, learning, evolutionary, language, knowledge, adaptive expression, gene, datum, cluster, analysis, microarray, factor, bind, classification, site market, stock, price, risk, financial, rate, evidence, return, exchange, work decision, make, agent, making, group, environment, autonomous, robot, human, mobile
ETM	network, neural, gene, expression, datum, cluster, classification, recurrent, learn, genetic - market, gas, price, stock, financial, natural, return, work, rate, estimate model, decision, base, analysis, method, theory, application, approach, make, dynamic
LDA	network, neural, learn, learning, recurrent, dynamic, model, artificial, sensor, bayesian gene, expression, datum, cluster, analysis, model, microarray, feature, sequence, base price, stock, oil, option, market, term, model, asset, return, pricing decision, theory, model, make, base, information, making, access, agent, bioinformatic

Table 5: Some related topics discovered by different topic models in the M10 corpus when run for $T = 10$ topics.

Model	Topics
CTM-Neg	neural, network, recurrent, language, artificial, grammatical, context, learn, symbolic, natural classification, neural, recognition, classifier, learn, pattern, coding, feature, network, sparse network, neural, recurrent, feedforward, artificial, genetic, bayesian, learn, knowledge, evolutionary
LDA	network, neural, recurrent, learn, *mechanic*, adaptive, inference, compute, *title*, *extraction*

Table 6: Some AI-related topics discovered by CTM-Neg and LDA in the M10 corpus when run for $T = 40$ topics. Italicized words in a topic appear less connected to the other words in the topic.

Therefore, we have kept the second line for ETM topics in Table 5 blank. We observed that some of the topics produced by ETM contain many common words. In particular, we found that five topics from ETM contain the words ‘model’, ‘decision’, ‘method’, ‘analysis’, and ‘theory’ in some order in the top slots, thus becoming repetitive, and consequently, ETM fails to discover meaningful and diverse topics like the other models. This is indicative of the component collapsing problem where all output topics are almost identical (Srivastava and Sutton, 2017).

We have observed earlier that on the M10 corpus, for large topic counts LDA beats CTM-Neg in IRBO but not in NPMI. We revisit this issue now and manually analyze their topics for $T = 40$. We found indeed the different topics output by LDA hardly overlap in words (leading to larger topic diversity) but the words do not always appear logically connected and interpretable (thus, sacrificing coherence). On the other hand, the topics generated by CTM-Neg look more coherent although they are not always disjoint. For example, see Table 6 which shows the topics containing the word ‘neural’ (among the top-10 words in the topic) discovered by CTM-Neg and LDA. CTM-Neg produces three topics that roughly relate to ‘natural language processing’, ‘pattern recognition’, and ‘neural and evolutionary computing’, respectively. But only one topic from LDA contains ‘neural’ – it is primarily about ‘neural networks’ but contains some very weakly related words.

6 Conclusion

We have proposed a negative sampling strategy for a neural contextualized topic model. We evaluated its performance on three publicly available datasets. In most of our experiments, the augmented model achieves higher topic coherence, as measured by NPMI and CV, and comparable topic diversity, as captured by IRBO, with respect to those of competitive topic models in the literature. A manual evaluation of a few selected topics shows that the topics generated by CTM-Neg are indeed coherent and diverse. In the future, we would like to compare it with other contrastive learning-based topic models and integrate it with other neural topic models.

Acknowledgments

This work is partially supported by the SERB-DST Project CRG/2021/000803 sponsored by the Department of Science and Technology, Government of India at Indian Association for the Cultivation of Science, Kolkata.

References

Suman Adhya and Debarshi Kumar Sanyal. 2022. What does the Indian Parliament discuss? an exploratory analysis of the question hour in the Lok Sabha. In *Proceedings of the LREC 2022 workshop on Natural Language Processing for Political Sciences*, pages 72–78, Marseille, France. European Language Resources Association.

Federico Bianchi, Silvia Terragni, and Dirk Hovy. 2021. Pre-training is a hot topic: Contextualized document embeddings improve topic coherence. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 759–766, Online. Association for Computational Linguistics.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022.

Jordan Boyd-Graber, Yuening Hu, David Mimno, et al. 2017. Applications of topic models. *Foundations and Trends® in Information Retrieval*, 11(2-3):143–296.

Adji B. Dieng, Francisco J. R. Ruiz, and David M. Blei. 2020. Topic modeling in embedding spaces. *Transactions of the Association for Computational Linguistics*, 8:439–453.

Alexander Hoyle, Pranav Goel, Andrew Hian-Cheong, Denis Peskov, Jordan Boyd-Graber, and Philip Resnik. 2021. Is automated topic model evaluation broken? the incoherence of coherence. *Advances in Neural Information Processing Systems*, 34:2018–2033.

Xuemeng Hu, Rui Wang, Deyu Zhou, and Yuxuan Xiong. 2020. Neural topic modeling with cycle-consistent adversarial training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9018–9030.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational Bayes. In *Proceedings of the International Conference on Learning Representations*.

Katsiaryna Krasnashchok and Salim Jouili. 2018. Improving topic quality by promoting named entities in topic modeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 247–253.

Jey Han Lau, David Newman, and Timothy Baldwin. 2014. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 530–539, Gothenburg, Sweden. Association for Computational Linguistics.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proceedings of the International Conference on Machine Learning*, pages 1727–1736. PMLR.

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.

David Newman, Youn Noh, Edmund Talley, Sarvnaz Karimi, and Timothy Baldwin. 2010. Evaluating topic models for digital libraries. In *Proceedings of the 10th Annual Joint Conference on Digital Libraries*, JCDL '10, page 215–224, New York, NY, USA. Association for Computing Machinery.

Thong Nguyen and Anh Tuan Luu. 2021. Contrastive learning for neural topic model. *Advances in Neural Information Processing Systems*, 34:11974–11986.

Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. 2016. Tri-party deep network representation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 1895–1901. AAAI Press.

Jipeng Qiang, Zhenyu Qian, Yun Li, Yunhao Yuan, and Xindong Wu. 2020. Short text topic modeling techniques, applications, and performance: a survey. *IEEE Transactions on Knowledge and Data Engineering*, 34(3):1427–1445.

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 and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the space of topic coherence measures. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 399–408.

Akash Srivastava and Charles Sutton. 2017. Autoencoding variational inference for topic models. In *Proceedings of the 5th International Conference on Learning Representations*.

Silvia Terragni, Elisabetta Fersini, Bruno Giovanni Galuzzi, Pietro Tropeano, and Antonio Candelieri. 2021. OCTIS: Comparing and optimizing topic models is simple! In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 263–270. Association for Computational Linguistics.

Rui Wang, Xuemeng Hu, Deyu Zhou, Yulan He, Yuxuan Xiong, Chenchen Ye, and Haiyang Xu. 2020. Neural topic modeling with bidirectional adversarial training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 340–350, Online. Association for Computational Linguistics.

Rui Wang, Deyu Zhou, and Yulan He. 2019. ATM: Adversarial-neural topic model. *Information Processing & Management*, 56(6):102098.

William Webber, Alistair Moffat, and Justin Zobel. 2010. A similarity measure for indefinite rankings. *ACM Transactions on Information Systems (TOIS)*, 28(4):1–38.

Xiaobao Wu, Chunping Li, Yan Zhu, and Yishu Miao. 2020. Short text topic modeling with topic distribution quantization and negative sampling decoder. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1772–1782, Online. Association for Computational Linguistics.

Lanling Xu, Jianxun Lian, Wayne Xin Zhao, Ming Gong, Linjun Shou, Daxin Jiang, Xing Xie, and Ji-Rong Wen. 2022. Negative sampling for contrastive representation learning: A review. *arXiv preprint arXiv:2206.00212*.

He Zhao, Dinh Phung, Viet Huynh, Yuan Jin, Lan Du, and Wray Buntine. 2021. Topic modelling meets deep neural networks: A survey. In *IJCAI 2021 Survey Track*.

A Appendix

A.1 Detailed Results of Quantitative Evaluation

Table 7 shows the NPMI, CV, and IRBO scores obtained for the different topic models on the three datasets for different topic counts. This table has been used to construct Table 2 and Fig. 2 in this paper.

Model	#Topics	Dataset: GN			Dataset: 20NG			Dataset: M10		
		NPMI	CV	IRBO	NPMI	CV	IRBO	NPMI	CV	IRBO
CTM-Neg	10	-0.096	0.439	1	0.1	0.641	0.997	0.073	0.491	1
CTM-Neg	20	0.031	0.424	1	0.129	0.672	0.994	0.076	0.49	0.993
CTM-Neg	30	0.13	0.494	1	0.135	0.664	0.992	0.065	0.475	0.989
CTM-Neg	40	0.174	0.542	0.999	0.129	0.656	0.991	0.06	0.461	0.986
CTM-Neg	50	0.201	0.562	0.998	0.132	0.662	0.99	0.051	0.461	0.985
CTM-Neg	60	0.227	0.592	0.998	0.125	0.65	0.991	0.046	0.449	0.982
CTM-Neg	90	0.235	0.598	0.995	0.114	0.626	0.988	0.031	0.44	0.979
CTM-Neg	120	0.234	0.592	0.993	0.107	0.616	0.986	0.019	0.426	0.976
CTM	10	-0.144	0.391	1	0.048	0.585	0.988	0.055	0.507	0.999
CTM	20	-0.01	0.406	0.998	0.105	0.649	0.994	0.06	0.467	0.988
CTM	30	0.076	0.467	0.997	0.111	0.661	0.992	0.069	0.479	0.983
CTM	40	0.126	0.515	0.996	0.104	0.644	0.991	0.051	0.466	0.98
CTM	50	0.129	0.51	0.994	0.101	0.636	0.99	0.044	0.456	0.977
CTM	60	0.149	0.519	0.993	0.095	0.628	0.99	0.042	0.452	0.974
CTM	90	0.153	0.535	0.991	0.094	0.608	0.989	0.04	0.456	0.971
CTM	120	0.17	0.538	0.99	0.086	0.606	0.986	0.025	0.443	0.968
ProdLDA	10	-0.103	0.431	1	0.069	0.602	0.986	0.047	0.477	0.999
ProdLDA	20	-0.007	0.403	1	0.09	0.645	0.991	0.017	0.45	0.992
ProdLDA	30	0.029	0.463	0.999	0.088	0.637	0.99	0.026	0.456	0.987
ProdLDA	40	0.081	0.491	0.997	0.087	0.63	0.993	0.035	0.458	0.982
ProdLDA	50	0.071	0.47	0.996	0.083	0.607	0.993	0.013	0.426	0.981
ProdLDA	60	0.098	0.481	0.995	0.085	0.607	0.991	0.028	0.447	0.979
ProdLDA	90	0.136	0.508	0.992	0.068	0.573	0.991	0.017	0.437	0.975
ProdLDA	120	0.14	0.518	0.99	0.068	0.57	0.99	0.02	0.436	0.969
ETM	10	-0.235	0.411	0.549	0.05	0.531	0.883	-0.004	0.325	0.653
ETM	20	-0.233	0.416	0.473	0.054	0.534	0.812	-0.07	0.346	0.525
ETM	30	-0.269	0.438	0.578	0.05	0.533	0.788	-0.063	0.333	0.449
ETM	40	-0.289	0.422	0.652	0.048	0.527	0.782	-0.061	0.353	0.462
ETM	50	-0.245	0.393	0.669	0.048	0.526	0.803	-0.069	0.35	0.523
ETM	60	-0.285	0.417	0.676	0.044	0.518	0.797	-0.058	0.351	0.437
ETM	90	-0.274	0.402	0.699	0.048	0.527	0.834	-0.056	0.351	0.464
ETM	120	-0.277	0.415	0.722	0.047	0.525	0.851	-0.069	0.353	0.505
LDA	10	-0.18	0.391	0.996	0.065	0.554	0.942	-0.035	0.389	0.966
LDA	20	-0.167	0.368	0.998	0.099	0.613	0.977	-0.133	0.373	0.986
LDA	30	-0.173	0.393	0.999	0.097	0.609	0.985	-0.183	0.374	0.991
LDA	40	-0.171	0.404	0.998	0.095	0.605	0.989	-0.218	0.389	0.993
LDA	50	-0.198	0.406	0.999	0.085	0.584	0.99	-0.251	0.402	0.995
LDA	60	-0.21	0.411	0.999	0.075	0.571	0.993	-0.261	0.408	0.994
LDA	90	-0.206	0.407	0.999	0.054	0.54	0.995	-0.274	0.398	0.994
LDA	120	-0.013	0.44	0.989	0.028	0.494	0.996	-0.184	0.352	0.99

Table 7: Performance of the different topic models on GN, 20NG, and M10 datasets for different topic counts. Each score is the median of 5 independent runs.

`IMFinE`: An Integrated BERT-CNN-BiGRU Model for Mental Health Detection in Financial Context on Textual Data

Ashraf Kamal **Padmapriya Mohankumar** **Vishal Kumar Singh**
`{ashrafkamal.mca,padmapriya.mohankumar,vishalsingh7x}@gmail.com`

Abstract

Nowadays, mental health is a global issue. It is a pervasive phenomenon over online social network platforms. It is observed in varied categories, such as *depression*, *suicide*, and *stress* on the Web. Hence, mental health detection problem is receiving continuous attention among computational linguistics researchers. On the other hand, public emotions and reactions play a significant role in financial domain and the issue of mental health is directly associated. In this paper, we propose a new study to detect mental health in financial context. It starts with two-step data filtration steps to prepare the mental health dataset in financial context. A new model called `IMFinE` is introduced. It consists of an input layer, followed by two relevant `BERT` embedding layers, a convolutional neural network, a bidirectional gated recurrent unit, and finally, dense and output layers. The empirical evaluation of the proposed model is performed on `Reddit` datasets and it shows impressive results in terms of *precision*, *recall*, and *f-score*. It also outperforms relevant state-of-the-art and baseline methods. To the best of our knowledge, this is the first study on mental health detection in financial context.

1 Introduction

The popularity of online social network (OSN) platforms, such as `Twitter`, `Facebook`, and `Reddit` have been growing at an unprecedented rate (Khan et al., 2022a). They have become a real-time and large-scale communication source to find and connect with users across the globe (Kamal and Abulaish, 2019a). These platforms offer users to express their moods, emotions, sentiments, and views. Besides that, users are involved in several activities like exchanging messages with friends, participating in ongoing trends, and establishing connection with celebrities. Hence, an enormous amount of data is generated from these online platforms (Abulaish and Kamal, 2018). It contains rich semantic information which can be used for several applications, such as *sentiment analysis* and *opinion mining*, *predictive modeling*, *Web surveillance*, *recommendation systems*, *information retrieval*, *data summarization*, and *cyber-security* (Khan et al., 2022b). In addition, it is also beneficial for several interdisciplinary studies including *psychology*, *behavioral*, and *cognitive* (Kamal and Abulaish, 2022).

In the last few years, it is seen that users consider OSNs to address mental health problems. Especially, after the occurrence of the *coronavirus disease* (`COVID-19`) pandemic, which has affected us at physical-, mental-, and psychological-level. The number of cases related to mental health problems, such as *depression*, *stress*, *anxiety*, and *suicide* have surged. As a result, it is a briefly discussed topic world-wide and rapidly increasing across OSNs in the form of *tweets*, *posts*, *comments*, and *blogs*. In the latest survey, the rate of increase of such mental health problems is found larger as compared to the physical health impacts in China (Huang and Zhao, 2020). Also, it is seen that around 80% of people reveal their intention of committing suicide on social media platforms (Sawhney et al., 2021). *Depression* causes frequent *stress* and it occurs due to many reasons, such as stressful life events including bereavement, divorce, prolonged illness, or financial issues. Further, prevailing it for a long span of time develops suicidal tendencies (Ansari et al., 2021).

Recently, people have been taking OSN platforms to communicate and receive advice on mental health-related issues. It has also motivated computational linguistic researchers in the sense that information mining from massive amount of user-generated contents (UGCs) can be used in mental health identification and detection. In this line, a wide range of natural language processing (NLP) and data mining approaches and techniques are applied along with handcrafted features for mental health classification on UGCs. Therefore, several traditional machine learning (ML) techniques and advanced deep learning (DL) models are taken into consideration for mental health classification (Ameer et al., 2022).

1.1 Mental Health and Finance

Financial stress is considered an economic determinant of mental health problems. It is directly linked with *depression* and *stress*. It spans world-wide, but mainly in those countries that have large populations and low-income (Alanazi et al., 2022). Existing literature has highlighted positive links between *depression* and several factors which lead to financial stress including debt, financial hardship, financial condition, economic situation, poverty, loan, mortgage, or low-income. Panic related to debt via loan like burdens may lead to sleeping disorder problem such as insomnia (Weissberger et al., 2020). It occurs due to the lack of sleep and affects the mental health of an individual. Similarly, other factors like low-income, mortgage, and poverty have affected an individuals' mental health which further leads to *depression* and *suicide* (Fitch et al., 2007). However, minimal attention is received in existing state-of-the-art (SOTA) in this direction of research as of now, especially using DL-based models. To this end, this study presents `IMFinE`, a new DL-based model to detect mental health in financial context.

1.2 Our Contributions

The role of context is crucial in text classification problems (Abulaish et al., 2020). In addition to that capturing semantic and syntactic information is also important for classification problems in the textual portion (Kamal and Abulaish, 2019b). Extracting mental health-related semantic and syntactic information via social media content is a challenging, interesting, and notable research problem, especially, when it is mentioned in a financial context. In this direction, this study presents a DL model for detecting mental health in financial context. Figure 1 presents the pipeline of the proposed work. It starts with data collection from `Reddit`, follow by data pre-processing, two-level data filtration steps to filter candidate mental health-based financial text, and finally a new DL-based model `IMFinE` is introduced which is responsible for binary classification task. `IMFinE` consists of an input layer, two parallel relevant `BERT` embeddings (`MentalBERT` (Ji et al., 2021) and `FinBERT` (Araci, 2019)), a convolutional neural network (CNN) (Kim, 2014), a bidirectional gated recurrent unit (BIGRU) (Liu and Guo, 2019), and end up with a dense and output layers.

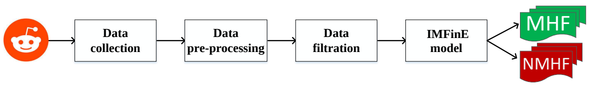


Figure 1: Pipeline of the proposed work

In, `IMFinE`, the input layer receives input either as a filtered candidate mental health-based financial (MHF) (refer as a positive class) text or normal non-mental health financial (NMFH) (refer as a negative class) text. Thereafter, it is passed to two parallel `BERT` embeddings (`MentalBERT` and `FinBERT`) to retrieve contextual information related to mental health and finance, respectively, and further it is concatenated to give comprehensive context representation. The role of the CNN is to receive that comprehensive context representation and obtain mental health and financial semantic and syntactic information, and the BiGRU layer is used to obtain preceding and succeeding mental health and financial information latent contextual information sequences present in the input text. Thereafter, it is forwarded to the dense and output layer, wherein *sigmoid* activation function is used to classify the input text as either MHF or NMHF.

Overall, the main contributions of this study are as follows:

- Exploring a novel mental health detection on financial textual data.
- Implementation of a two-steps dataset filtration technique to identify candidate mental health-related financial texts.
- Development of a new DL-based `IMFinE` model to detect mental health in financial context.
- Conducting an empirical evaluation of the proposed model and compared with SOTA and baselines methods to examine its efficacy.

The rest of the paper is organized as follows. Section 2 presents a brief review of the existing works. Section 3 highlights the problem description and provides insights about the dataset preparation.

Section 4 presents the architectural work-flow of the proposed model. Section 5 demonstrates the experimental setup and evaluation results. Finally, Section 6 concludes the paper and highlights future directions of research.

2 Related Work

This section presents the existing SOTA related to mental health identification and detection on OSN platforms. Besides that, this section also highlights the current status and limitations in the end.

Park et al. (2012) analyzed the language which indicates depressive moods on `Twitter` data. They considered depressive attitudes and actions of users, organized and conducted interviews between two users, and analyzed the correlation between interviews and data available on `Twitter`. Choudhury et al. (2013) highlighted "major depressive disorders" prediction and considered many behavioral patterns of depressed users from this social media data. Tsugawa et al. (2015) considered users behavior to predict *depression* on `Twitter` in Japanese language. Ronghua and Qingpeng (2016) explained that users reveal their moods related to depression analysis via social media. Shen et al. (2017) introduced six depression-based feature groups. They proposed a multi-modal *depressive* dictionary learning model for depressed user detection on `Twitter`. Haque et al. (2021) identified *suicide* and *depression* via DL techniques. Xue et al. (2014) proposed stress detection on `Twitter`. They analyzed psychological pressures in teenagers' tweets. Lin et al. (2016) extracted semantic features and combined multi-task learning using CNN for identification of *stress* related topics and events on social media data. Thelwall (2017) introduced a rule-based approach to analyze *stress* and relaxation using both direct and indirect expressions on `Twitter`. Turcan and McKeown (2019) considered `Reddit` and proposed a dataset namely, `Dreadit` for stress classification. They applied a manual approach to construct this dataset. Coppersmith et al. (2016) considered posts by users before the suicide attempt on `Twitter` and analyzed the lexical markers and emotions in it. Giannakakis et al. (2017) considered facial clues from the recorded video for *stress* and *anxiety* detection using ML techniques. Li and Liu (2020) presented stress detection using DL techniques. They applied CNN and multi-layer perceptron. Rastogi et al. (2022) proposed a new dataset for *stress* detection on `Twitter` and `Reddit` datasets.

All aforementioned literature presents the availability of the datasets across OSN platforms and highlights important insights for mental health detection in terms of *depression*, *suicide*, and *stress*. However, there is no literature that addresses mental health detection in financial context using textual data, which is a challenging, worth, and significant research investigation task. To the best of our knowledge, this is the first study on mental health detection in financial context on textual data via DL-based models.

3 Problem Description and Dataset Preparation

This section presents the problem description for mental health detection in finance and covers dataset preparation.

3.1 Problem Description

This study presents the mental health detection in finance on textual data. The considered problem represents a binary classification problem. A piece of textual data is classified as either MHF or NMHF. In this study, we consider three commonly found mental health-related categories – *depression*, *suicide*, and *stress* associated to financial textual data over social media content. A formal definition of each mental health categories is given below followed by a relevant example taken from the `Reddit` posts.

- *Depression*: It is defined as a "*medical condition in which a person feels very sad, anxious and without hope and often has physical symptoms such as being unable to sleep, etc*".[1] **For example**:"*Debts and suicidal thoughts. During the year of 2021 I've Lost* 3 *relatives and got a debt of* 2000 *USD due to that. I'm getting more and more depressed and i'm not sleeping well...*".

- *Suicide*: It is defined as "*a death that happens when someone harms themselves because they want to end their life. It is one of the mental health problems and a leading cause of death.*"[2] **For example**: "Being broke makes me want to kill myself Bills keep coming.

[1] `https://bit.ly/3LJ3R16`
[2] `https://medlineplus.gov/suicide.html`

I don't have enough money for anything. Saving change just to eat...".

- *Stress*: It is defined as "*any type of change that causes physical, emotional, or psychological strain*".[3] **For example**: "I'm having bad thoughts...I'm about to lose my house and no where to go. I'm 4 months behind on my mortgage (got laid off from my job because of covid)...".

3.2 Dataset Preparation

This sub-section presents the dataset collection and pre-processing, and filtration of candidate mental health financial text from these datasets.

3.2.1 Dataset Collection and Pre-processing

In this study, `Reddit` posts are taken to prepare the dataset. We have used `Reddit` API[4] using `PRAW`[5] wrapper to retrieve posts based on mental health and non-mental health. All posts are collected via *subreddits*, which presents specific topic on `Reddit` and preceded by r/. We have used *subreddits* based on mental health, such as r/mentalhealth, r/mmfb (make me feel better), r/offmychest, r/traumatoolbox, r/anxietyhelp, r/CPTSD (complex post traumatic stress disorder), r/depression, and r/SuicideWatch to prepare mental health datasets. Likewise, *subreddits*, such as r/happy, r/mademesmile, r/makemesmile, r/financeadvice, r/creditcards, r/wholesome, r/economics, r/financialindependence, r/financialplanning, r/investing, r/personalfinance, r/pftools, and r/tax are considered to prepare a non-mental health datasets. Thus, 320000 mental health instances and 270000 non-mental health instances are finalized. Further details about the datasets are given in the sub-section 5.1.

All collected raw datasets consist of many noise and non-literal information. Removal of such unwanted information is crucial to achieve good classification accuracy. Considering this, we have performed data cleaning steps and removed symbols, hexa characters, mentions, hashtags, slash, exclamation, quotation, and punctuation marks, unwanted links, ampersands, extra dots and spaces, digits, and non-ASCII characters. In the end, we have converted raw text into the lower-case form.

3.2.2 Dataset Filtration

On analyzing all mental health datasets, it is found that all mental health instances do not contain financial information. Also, there is no availability of benchmark datasets on financial mental health. Considering this, two-steps filtering technique is applied to filter only those mental health instances from the datasets, which are context- and semantic-wise related to finance.

To this end, in step-1, we have compiled a list of keywords based on unigram and bigram tokens from (Guan et al., 2022) work along with our generated keywords (i.e, unigram and bigram) tokens related to finance or financial situation, as given in table 1. We filter candidate mental health instances using regular expression-based criteria, in which if at least one of the keywords (unigram/bigram) matches with pre-processed mental health instance from table 1.

Table 1: A List of keywords based on finance or financial situation

List of keywords (unigram and bigram)
'income', 'debt', 'loan', 'mortgage', 'finance', 'economy', 'job', 'broke', 'poor', 'poverty', 'homeless', 'salary', 'money', 'bank', 'savings', 'scam', 'robbery', 'deprivation', 'loss', 'fund', 'earn'', 'payroll'', 'earning', 'wage', 'fired', 'livelihood', 'compensation', 'revenue', 'allowance', 'payoff', 'wealth', 'asset', 'economic situation', 'economic stat', 'economic condition', 'economic position', 'economic hardship', 'economic str', 'economic difficult', 'financial situation', 'financial position', 'financial condition', 'financial stat', 'financial hardship', financial difficult'

Thereafter, in step-2, we have applied zero-shot learning (ZSL), which aims to perform predictions without having seen labelled training instances. It is widely used in NLP and text classification problems, in which combination of seen and unseen labelled via auxiliary information are taken to encode the available discrimination attributes of an instances (Xian et al., 2018). `Facebook/Bart-large-mnli`[6] is one of the popular ZSL method which is based on `Hugging`

[3]`https://bit.ly/2yYHnVu`
[4]`https://www.reddit.com/dev/api/`
[5]`https://bit.ly/3r7690r`

[6]`https://huggingface.co/facebook/bart-large-mnli`

Face-based Transformer model trained on multi natural language interface (*aka*, MNLI) dataset. It is based on two concepts – *premises* (refers to the instances which is to be classified) and *hypothesis* (refers to the number of class labels) (Plaza-Del-Arco et al., 2022). A confidence score is generated for each *hypothesis* of a given *premise*, and based on the highest confidence score, the *premise* is predicted. In this study, we set threshold of 0.89 for the highest confidence score.

Considering this, each filtered mental health instance in step-1 is passed as a *premise* and accordingly two *hypothesis* are taken - *financial* and *non-financial* for labelling purpose. As a result, candidate financial mental health instances are collected based on the highest confidence score. Figure 2 presents an example based on the *premise* and *hypothesis* concepts related to MNLI to ZSL methods. In this example, the representation of finance related mental health is *entailed*. Hence, it is predicted as MFH instance.

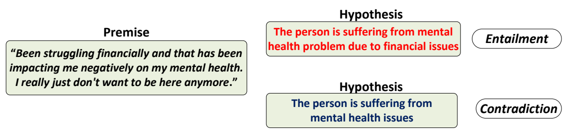
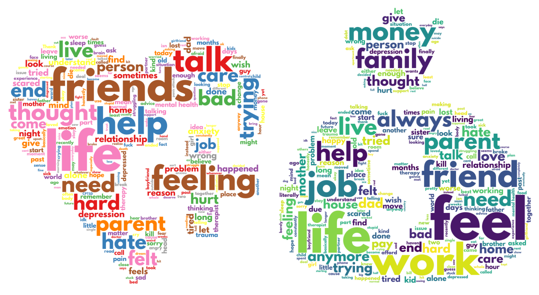

Premise
"Been struggling financially and that has been impacting me negatively on my mental health. I really just don't want to be here anymore."
Hypothesis
The person is suffering from mental health problem due to financial issues
Entailment
Hypothesis
The person is suffering from mental health issues
Contradiction

Figure 2: An example for data filtration using MNLI to ZSL methods for mental health instance.

Figure 3 presents a better visual insights and effectiveness of data filtration step on collected Reddit posts via two word clouds. It can be seen from this figure that only frequent words related to mental health are seen before data filtration step, whereas top-frequent words related to finance and mental health are found after data filtration step.



(a) Before data filtration (b) After data filtration

Figure 3: Effect of data filtration step via word cloud

4 Proposed Model

This section presents a detailed description of the proposed IMFinE model. It includes an input layer, two embeddings (MentalBERT and FinBERT), CNN, BiGRU, followed by a dense and output layers. Figure 4 presents the architecture of the proposed IMFinE model.

4.1 Input Layer

Given the candidate MHF or NMHF as input text n consisting of w_n words, the input layer tokenizes each word available in its textual contents. Each token is indexed in a dictionary and converted into a numeric vector v such that it replaces according to its index value as per the dictionary. The length of v is different because of the varying input length size. Hence, a fixed-length l of input vectors is maintained for each input text. Thus, v is transformed to a padded-vector p of 200 fixed-length, such that $|p| = l \geq |v|$. The fixed-length resulting vector $p \in R^{1\times l}$ is then forwarded to the embedding layer.

4.2 Embedding Layers

In the last few years, BERT has gained immense popularity because it deals with contextual information in both directions via Transformer. MentalBERT is a pre-trained masked language model using BERT for mental health detection tasks, whereas FinBERT is a pre-trained NLP model and it is built by training the BERT model in the financial domain over huge financial data. Considering this, we leverage these two specific BERT models – MentalBERT and FinBERT based on our problem statement to retrieve contextualized language representations based on mental health and finance from the input text.

The generated input vector from the input layer is passed to both parallel embedding layers with a maximum sequence length of 200. We consider 768-dimensional word vector representation for both MentalBERT and FinBERT embeddings. Consequently, the relevant contextual information based on mental health and finance is retrieved from both embeddings in parallel. The encoded representation from two embeddings is concatenated and that gives a rich comprehensive contextual representation of the input text. Further, it is passed to the CNN layer.

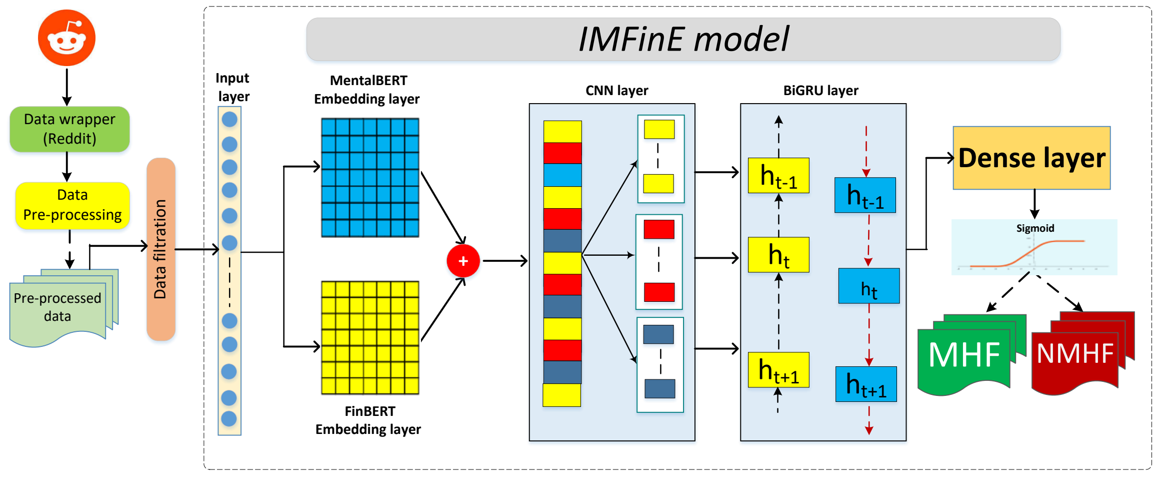


Figure 4: Architecture of the proposed IMFinE model

4.3 CNN Layer

Kim (2014) proposed CNN which extracts low-level semantic and syntactic features automatically. It extracts contextual local and positional invariant features, performs several convolution operations on the received input texts, and generates the global feature vector. In this study, we have used 64 filters of window size 3 that moves the comprehensive embedding vectors for extracting features related to mental health and finance. Max pooling operation of pool size 3, ReLU activation function, and 64 filters perform the convolutional operation from top to bottom, and extract the feature sequence as $f_t = [f_1, f_2, ..., f_{64}]$ based on mental health and finance for the input text. The n^{th} feature sequence, f_t from word window x_t is generated, as given in 1. Finally, the filter outputs are concatenated to give the resultant mental health and finance-based feature representation, which is forwarded to the BiGRU layer.

$$f_t = r(w_t \cdot x_t + b) \tag{1}$$

4.4 BiGRU Layer

Liu and Guo (2019) proposed GRU, which deals with long-term temporal dependencies without dealing the vanishing gradient descent problem. It consists of two gates and operational in both forward and backward directions as forward GRU and backward GRU, respectively. In this study, the role of the forward GRU and backward GRU is to generate succeeding feature sequences (f_t to f_{64}) and preceding feature sequences (f_{64} to f_t), respectively for mental health and finance related latent contextual feature sequences from the extracted features of the CNN layer. Thus, equations 2 and 3 show forward and backward directions of BiGRU outcomes, respectively. The resultant combined outcome of both forward and backward directions is passed to the dense layer.

$$\overrightarrow{gru_f} = \overrightarrow{GRU}(L_{f_t}), n \in [1, 64] \tag{2}$$

$$\overleftarrow{gru_b} = \overleftarrow{GRU}(L_{f_t}), n \in [64, 1] \tag{3}$$

4.5 Dense and Output Layers

In this study, both datasets are divided as a training set, a testing set, and a validation set, wherein 70% is used for training, 20% is used for testing, and 10% is used for validation, we use 30 epochs with early stopping and Adam as an optimization algorithm. The fully connected dense layer gives features set based on the outcome of previous layers divisible into two classes. Finally, the *sigmoid* activation function is used upon the dense layer, and *binary cross-entropy* loss function is used for classifier training which interprets input text labelled as MHF or NMHF.

5 Experimental Setup and Results

This section presents the experimental details of the proposed IMFinE model. It includes the statistical details of the datasets, experimental and hyper-parameter settings, evaluation metrics, followed by evaluation results and comparative analysis, and ablation study.

5.1 Datasets

This section presents the datasets used in this study. We have considered three datasets from `Reddit`, out of which two are benchmark datasets. A short description of the two benchmark datasets is given below:

- **Dreddit dataset**: Turcan and McKeown (2019) proposed this dataset, where they collected 1857 *stress* and 1696 *non-stress* instances to prepare training and testing datasets.
- **SDCNL dataset**: Haque et al. (2021) proposed this dataset, where they collected 915 *depression* and 981 *suicide* instances to prepare training and testing datasets.

We have combined *stress*, *depression*, and *suicide* instances of both benchmark datasets for mental health category and *non-stress* instances as non-mental health category. As a result, a total number of 3753 and 1696 instances is finalized for mental health and non-mental health categories, respectively after combining both benchmark datasets and named it as DS-1. Besides that, we have prepared one dataset using `Reddit`, as discussed in sub-section 3.2.1, and named it as DS-2. Further, we have applied dataset filtration techniques across all datasets as discussed in the sub-section 3.2.2. Table 2 presents the final statistics of the datasets after data filtration steps.

Table 2: Final statistics of the datasets after data filtration steps

Datasets ↓	MHF	NMHF	Total
DS-1	509	2613	3122
DS-2	8400	8400	16800

5.2 Experimental and Hyperparameter Settings

This section presents the details about the experimental and hyperparameter settings used for the implementation of the proposed `IMFinE` model. Table 3 presents the summary of the experimental settings. Likewise, table 4 presents the summary of the hyperparameter settings.

5.3 Evaluation Metrics

The proposed model is evaluated using four standard evaluation metrics – *precision*, *recall*, *f-score*, and *accuracy*. Equations 4 to 7 define these metrics formally using the terms – *true positive* (*TP*), *false positive* (*FP*), *true negative* (TN), and *false negative* (FN). TP represents the total number of correctly classified MHF instances. FP represents the total number of incorrectly classified MHF instances. TN represents the total number of correctly classified NMHF instances. Finally, FN represents the total number of incorrectly classified NMHF instances.

Table 3: A summary of the experimental settings

Environment	Configurations
Machine	`Intel Haswell`
Operating system	`Ubuntu-20.04`
Memory (RAM)	32 GB
GPU	`NVIDIA Tesla`
Language	`Python 3.9`
Neural network library	`Keras 2.10.0`
`Reddit` API wrapper	`PRAW`

Table 4: A summary of the hyperparameter settings

Hyperparameter	Values
Batchsize	32
Padding	200
Spatial dropout	0.2
Dropout	0.2
#neurons (GRU)	100
#filters (CNN)	64
window size (CNN)	3
pool size (CNN)	3

$$Precision\ (P) = \frac{TP}{TP + FP} \tag{4}$$

$$Recall\ (R) = \frac{TP}{TP + FN} \tag{5}$$

$$F\text{-}score\ (F) = \frac{2 \times P \times R}{P + R} \tag{6}$$

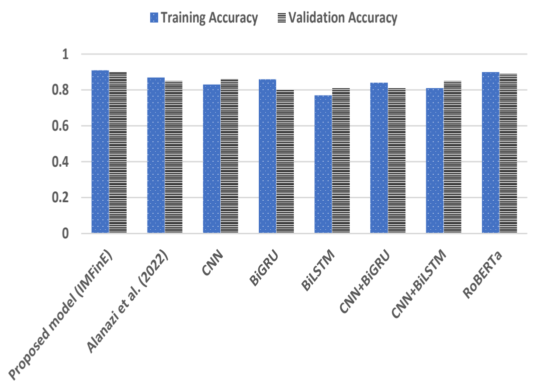
$$Accuracy\ (A) = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

5.4 Evaluation Results and Comparative Analysis

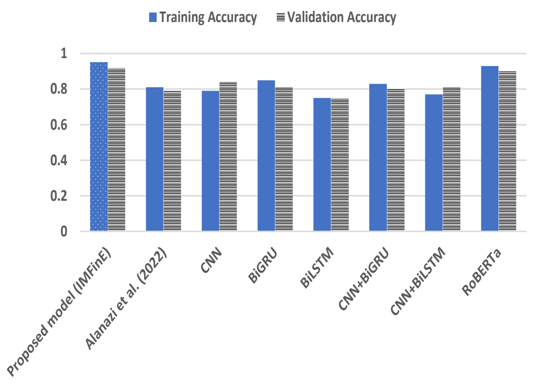
In this section, we present the evaluation results of the proposed `IMFinE` model and compared it with a recent study proposed by Alanazi et al. (2022), wherein authors addressed

Table 5: Performance results on both datasets

Datasets →	**DS-1**			**DS-2**		
Methods ↓	**P**	**R**	**F**	**P**	**R**	**F**
Proposed model (`IMFinE`)	**0.82**	**0.86**	**0.84**	**0.95**	**0.93**	**0.94**
Alanazi et al. (2022)	0.74	0.80	0.77	0.91	0.92	0.91
CNN	0.72	0.83	0.77	0.90	0.89	0.89
BiGRU	0.80	0.84	0.82	0.93	0.91	0.92
BiLSTM	0.76	0.83	0.79	0.91	0.91	0.91
CNN+BiGRU	0.70	0.84	0.76	0.90	0.88	0.89
CNN+BiLSTM	0.67	0.81	0.73	0.88	0.90	0.89
`RoBERTa`	0.81	0.85	0.83	0.94	0.85	0.89



(a) DS-1 dataset



(b) DS-2 dataset

Figure 5: Training versus validation accuracy on DS-1 and DS-2 datasets

mental health monitoring using sentimental analysis of financial text. We have also compared our proposed `IMFinE` model with neural network-based baseline methods. Table 5 presents the performance evaluation results on both datasets. It can be observed that proposed `IMFinE` model shows good result on both datasets. It receives *f-score* of 0.84 and 0.94 for DS-1 and DS-2, respectively. This refers to an interesting observation that the proposed model is performing better on both unbalanced (DS-1) and balanced (DS-2) datasets. It can also be observed that it performs significantly better than SOTA work and neural network baseline methods in terms of *precision*, *recall*, and *f-score*. However, BiGRU shows the best performance across neural network baseline models, but our proposed model is performing more better and leveraging contextual information from both embedding on both datasets and receiving latent contextual information sequences. One more interesting observation is the proposed `IMFinE` model performs better with smaller dataset (i.e., DS-1) as well.

Figure 5 presents the visualization of the training versus validation accuracy on both datasets. It shows that the proposed model performs impressive in terms of training versus validation accuracy on DS-1 and DS-2 datasets. It can also be seen that the proposed model shows significantly better results in comparison to the SOTA and neural network baseline results.

Table 6: Ablation study of the proposed model in terms of *f-score* on DS-1 and DS-2 datasets. datasets.

Proposed model (`IMFine`)	**DS-1**	**DS-1**
All layers	**0.84**	**0.94**
Without CNN	0.80	0.91
Without BiGRU	0.81	0.90

5.5 Ablation Study

This section presents the ablation study of the proposed model to show component-wise analysis.

Table 6 shows the proposed model with all layers performs better. The performance of the model is effected without considering CNN and BiGRU.

6 Conclusion

We have proposed a new problem for mental health detection in financial context. It has started with two-level data filtration steps to filter candidate mental health instances in financial domain. We have proposed a novel `IMFinE` model which is consisted of input, `BERT`-based relavant contextual embeddings (`MentalBERT` and `FinBERT`), CNN, BiGRU, and, output followed dense layers. The proposed `IMFinE` model has evaluated over `Reddit` datasets and experimental results have shown significantly better results in comparison to SOTA and several neural network baseline methods. The evaluation of the `IMFinE` model in a multi-modal setting could be an important direction of research. `IMFinE` model can also be extended over multi-lingual datasets.

References

Muhammad Abulaish and Ashraf Kamal. 2018. Self-deprecating sarcasm detection: An amalgamation of rule-based and machine learning approach. In *Proceeding IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 574–579. IEEE.

Muhammad Abulaish, Ashraf Kamal, and Mohammed J. Zaki. 2020. A survey of figurative language and its computational detection in online social networks. *ACM Transactions on the Web (TWEB)*, 14(1):1–52.

Saad A. Alanazi, Ayesha Khaliq, Fahad Ahmad, Nasser Alshammari, Iftikhar Hussain, Muhammad A. Zia, Madallah Alruwaili, Alanazi Rayan, Ahmed Alsayat, and Salman Afsar. 2022. Public's mental health monitoring via sentimental analysis of financial text using machine learning techniques. *International Journal of Environmental Research and Public Health*, 19(15):9695.

Iqra Ameer, Muhammad Arif, Grigori Sidorov, Helena Gòmez-Adorno, and Alexander Gelbukh. 2022. Mental illness classification on social media texts using deep learning and transfer learning. *arXiv preprint arXiv:2207.01012*.

Gunjan Ansari, Muskan Garg, and Chandni Saxena. 2021. Data augmentation for mental health classification on social media. In *Proceedings of the Sixteen International Conference on Natural Language Processing*, pages 152–161, National Institute of Technology, Silchar, India. NLP Association of India.

Dogu Araci. 2019. Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*.

Munmun D. Choudhury, Michael Gamon, Scott Counts, and Eric Horvitz. 2013. Predicting depression via social media. In *Proceeding the Seventh International AAAI Conference on Weblogs and Social Media*.

Glen Coppersmith, Kim Ngo, Ryan Leary, and Anthony Wood. 2016. Exploratory analysis of social media prior to a suicide attempt. In *Proceedings of the third workshop on computational linguistics and clinical psychology*, pages 106–117.

Chris Fitch, Robert Chaplin, Colin Trend, and Sharon Collard. 2007. Debt and mental health: the role of psychiatrists. *Advances in Psychiatric Treatment*, 13(3):194–202.

Giorgos Giannakakis, Matthew Pediaditis, Dimitris Manousos, Eleni Kazantzaki, Franco Chiarugi, Panagiotis G. Simos, Kostas Marias, and Manolis Tsiknakis. 2017. Stress and anxiety detection using facial cues from videos. *Biomedical Signal Processing and Control*, 31:89–101.

Naijie Guan, Alessandra Guariglia, Patrick Moore, Fangzhou Xu, and Hareth Al-Janabi. 2022. Financial stress and depression in adults: A systematic review. *PloS one*, 17(2):e0264041.

Ayaan Haque, Viraaj Reddi, and Tyler Giallanza. 2021. Deep learning for suicide and depression identification with unsupervised label correction. In *Proceedings of the Thirtieth International Conference on Artificial Neural Networks, Bratislava, Slovakia, 2021*, pages 436–447.

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.

Shaoxiong Ji, Tianlin Zhang, Luna Ansari, Jie Fu, Prayag Tiwari, and Erik Cambria. 2021. Mentalbert: Publicly available pretrained language models for mental healthcare. *arXiv preprint arXiv:2110.15621*.

Ashraf Kamal and Muhammad Abulaish. 2019a. An lstm-based deep learning approach for detecting self-deprecating sarcasm in textual data. In *Proceedings of the 18th International Conference on Natural Language Processing*, pages 201–210, International Institute of Information Technology, Hyderabad, India. NLP Association of India.

Ashraf Kamal and Muhammad Abulaish. 2019b. Self-deprecating humor detection: A machine learning approach. In *International Conference of the Pacific Association for Computational Linguistics*, pages 483–494. Springer.

Ashraf Kamal and Muhammad Abulaish. 2022. Cat-bigru: Convolution and attention with bi-directional gated recurrent unit for self-deprecating sarcasm detection. *Cognitive Computation*, 14(1):91–109.

Shakir Khan, Mohd Fazil, Vineet K. Sejwal, Mohammed A. Alshara, Reemiah M. Alotaibi, Ashraf Kamal, and Abdul R. Baig. 2022a. Bichat: Bilstm with deep cnn and hierarchical attention for hate speech detection. *Journal of King Saud University-Computer and Information Sciences*, 34(7):4335–4344.

Shakir Khan, Ashraf Kamal, Mohd Fazil, Mohammed A. Alshara, Vineet K. Sejwal, Reemiah M. Alotaibi, Abdul R. Baig, and Salihah Alqahtani. 2022b. Hcovbi-caps: Hate speech detection using convolutional and bi-directional gated recurrent unit with capsule network. *IEEE Access*, 10:7881–7894.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Russell Li and Zhandong Liu. 2020. Stress detection using deep neural networks. *BMC Medical Informatics and Decision Making*, 20(11):1–10.

Huijie Lin, Jia Jia, Liqiang Nie, Guangyao Shen, and Tat-Seng Chua. 2016. What does social media say about your stress?. In *IJCAI*, pages 3775–3781.

Gang Liu and Jiabao Guo. 2019. Bidirectional lstm with attention mechanism and convolutional layer for text classification. *Neurocomputing*, 337:325–338.

Minsu Park, Chiyoung Cha, and Meeyoung Cha. 2012. Depressive moods of users portrayed in twitter. In *Proceedings of the Eighteen ACM International Conference on Knowledge Discovery and Data Mining, SIGKDD 2012*, pages 1–8.

Flor Miriam Plaza-Del-Arco, María-Teresa Martín-Valdivia, and Romann Klinger. 2022. Natural language inference prompts for zero-shot emotion classification in text across corpora. *arXiv preprint arXiv:2209.06701*.

Aryan Rastogi, Liu Qian, and Erik Cambria. 2022. Stress detection from social media articles: New dataset benchmark and analytical study. In *Proceeding of the IEEE World Conference of Computational Intelligence*.

XU Ronghua and Zhang Qingpeng. 2016. Understanding online health groups for depression: Social network and linguistic perspectives. *Journal of medical Internet research*, 18(3):e5042.

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.

Guangyao Shen, Jia Jia, Liqiang Nie, Fuli Feng, Cunjun Zhang, Tianrui Hu, Tat-Seng Chua, and Wenwu Zhu. 2017. Depression detection via harvesting social media: A multimodal dictionary learning solution. In *IJCAI*, pages 3838–3844.

Mike Thelwall. 2017. Tensistrength: Stress and relaxation magnitude detection for social media texts. *Information Processing & Management*, 53(1):106–121.

Sho Tsugawa, Yusuke Kikuchi, Fumio Kishino, Kosuke Nakajima, Yuichi Itoh, and Hiroyuki Ohsaki. 2015. Recognizing depression from twitter activity. In *Proceedings of the Thirty Third annual ACM conference on human factors in computing systems*, pages 3187–3196.

Elsbeth Turcan and Kathleen McKeown. 2019. Dreaddit: A reddit dataset for stress analysis in social media. *EMNLP-IJCNLP 2019*, page 97.

Gali H Weissberger, Laura Mosqueda, Annie L Nguyen, Anya Samek, Patricia A Boyle, Nguyen Caroline P, and S Duke Han. 2020. Physical and mental health correlates of perceived financial exploitation in older adults: Preliminary findings from the finance, cognition, and health in elders study (finches). *Aging & mental health*, 24(5):740–746.

Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. 2018. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2251–2265.

Yuanyuan Xue, Qi Li, Li Jin, Ling Feng, David A. Clifton, and Gari D. Clifford. 2014. Detecting adolescent psychological pressures from micro-blog. In *International Conference on Health Information Science*, pages 83–94. Springer.

Methods to Optimize *Wav2Vec* with Language Model for Automatic Speech Recognition in Resource Constrained Environment

Vaibhav Haswani
vaibhavhaswani@gmail.com

Padmapriya Mohankumar
padmapriya.mohankumar@gmail.com

Abstract

Automatic Speech Recognition (ASR) on resource constrained environment is a complex task since most of the State-Of-The-Art models are combination of multilayered convolutional neural network (CNN) and Transformer models which itself requires huge resources such as GPU or TPU for training as well as inference. The accuracy as a performance metric of an ASR system depends upon the efficiency of phonemes to word translation of an Acoustic Model and context correction of the Language model. However, inference as a performance metric is also an important aspect, which mostly depends upon the resources. Also, most of the ASR models uses transformer models at its core and one caveat of transformers is that it usually has a finite amount of sequence length it can handle. Either because it uses position encodings or simply because the cost of attention in transformers is actually O(n^2) in sequence length, meaning that using very large sequence length explodes in complexity/memory. So you cannot run the system with finite hardware even a very high-end GPU, because if we inference even a one hour long audio with Wav2Vec the system will crash. In this paper, we used some state-of-the-art methods to optimize the Wav2Vec model for better accuracy of predictions in resource constrained systems. In addition, we have performed tests with other SOTA models such as Citrinet and Quartznet for the comparative analysis.

1 Introduction

Speech is the most natural way of human communication; it gives humans a medium to understand and communicate their feelings and emotions. Understanding speech or speech recognition is a critical part of modern applications even plays a significant role for empowering AI enabled smart devices such as Amazon's Echo, Google's Nest, Apple's Homepod etc.
In this paper, we have proposed a method called `WSLR` (Wav2Vec with Stride Chunking and Language Model for Resource-constrained devices) to create an offline speech recognition system using Wav2Vec model which achieved a Word Error Rate (WER) of 0.85 in an environment of 2 core CPU and 4 gigabytes of RAM on Librispeech test set, which surpasses the score of original Wav2Vec model i.e. 1.8 and other SOTA models. To recognize speech of an audio file we first start off by converting it into a digital format since audio data could have a lot of variations like different channels or sample rate. We further have must standardize the dataset by re-sampling the data to a specific sample rate as any model requires data to follow common standards. We have used Wav2Vec-base model trained on `librispeech` data for 960 hours as our main ASR model for the speech recognition task where it encodes sound files via multi-layer convolutional neural network (CNN) to produce latent speech representation and then feed those masked representations to a transformer network, the output of which can be decoded to word vectors using a Connectionist Temporal Classification (CTC) algorithm. The following model can further be optimized when the output logits are fed to a language model (in this paper we used a 4-gram language model). (Baevski et al., 2020) [1]
To deploy our model to a resource constrained environment we use something called *stride chunking*, here the chunking is sequential, and we chunk the audios such that there is some overlapping length at borders defined by 'stride length' and the main context stay intact within the center. Then while inference, overlapped segments or logits (as output) are dropped to keep the results of the transcriptions as similar as they would be while inferenced with full audio. We also use some rule-based post-processing steps for our transcription to generalize even better for spoken numbers or symbols in-case.

[1]stride chunking reference at https://huggingface.co/blog/asr-chunking

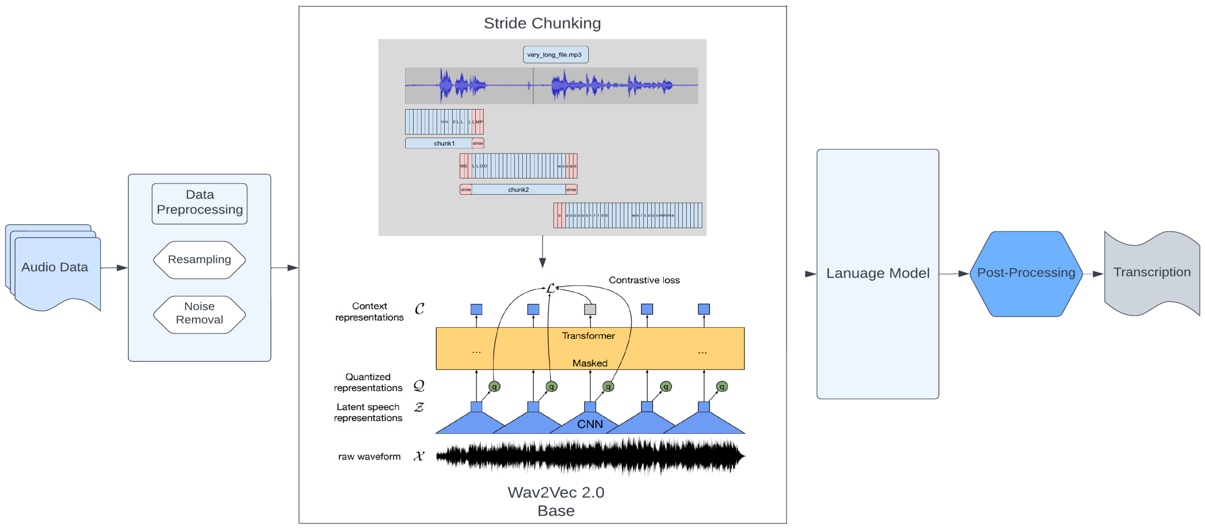


Figure 1: Proposed workflow of optimized Wav2Vec

2 Proposed Method

In this section, we discuss the proposed approach i.e. WSLR to optimize the state-of-the-art wav2vec model introduced by Facebook and that achieves such an optimal performance when inferenced in constrained environments such as, constrained docker container or embedded/micro systems like Raspberry Pi or Nvidia's Jetson board. Constrained inference along with good scores is quite a big deal for a huge deep learning model. Since, more the size of the model the more resources it demands and Wav2Vec is itself a combination of huge models such as CNNs and Transformers. Therefore, it is quite a challenging task. The slight drop in scores while chunking is comprehended with pre-processing, language model, and post-processing optimizations. Further, the details of the optimization methods and proposed functionalities are discussed in the following sub-sections.

2.1 Data Processing

Data Processing of audios aims to standardize the audios with a uniform setting, for evaluations we have used `librispeech` test-clean set by (Panayotov et al., 2015), and for explicit data pre-processing we have used `ffmpeg` utility for conversion of media to standard .wav format and `librosa` package for loading and re-sampling files. We first off defined pre-processing module to convert files to standard wav format and then re-sample it to 16KHz since Wav2Vec 2.0 strictly targets audios sampled at the mentioned rate. Text pre-processing methods such as, lower-casing, removal of special case characters and unwanted white spaces are performed on the librispeech test transcripts.

2.2 Stride Chunking

One way to inference from a transformer-based model for long audios in constrained environment could be by chunking the audio in parts of equal length (Mauranen and Vetchinnikova, 2017) and send those chunks to the model iteratively and later generate the whole prediction by aggregating all the sub predictions. This is computationally efficient but usually leads to subpar results. A major caveat of this kind of chunking is poor context at the border of chunks, i.e., since model requires previous context to generate better transcriptions this kind of chunking would give a non-contextual part of speech to the model which would lead to poor transcription.

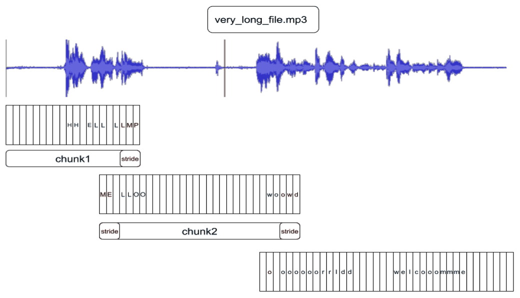


Figure 2: Reference for stride chunking from huggingface

Stride Chunking is even a smarter way of chunking audios inspired by stride concept in CNNs where we chunk the audios by leaving some overlapping frames from previous and next chunk, this way model will have proper context in the center for the inference. In stride chunking, while inference overlapped segments or logits are dropped after inference to generate aggregated output of all chunks i.e. proper transcription as we get it from the whole audio file. In our experiments we have tested different hyper-parameters and found that 30 seconds of `chunk length` with 2 seconds of `stride length` from right and 1 second of stride from left are giving optimal results in terms of inference and recognition quality.

2.3 ASR Model - Wav2Vec Base 960h

As an ASR model we used state-of-the-art model by Facebook-AI i.e., Wav2Vec 2.0 base 960h (we also used Wav2Vec 2.0 large in our experiments but selected the base model since it performs better for constrained environments). Proposed in (Baevski et al., 2020), “Wav2Vec is composed of a multi-layer convolutional feature encoder f : X $\rightarrow$ Z which takes as input raw audio X and outputs latent speech representations z1, . . . , zT for T time-steps. They are then fed to a Transformer g: Z $\rightarrow$ C to build representations c1, . . . , cT capturing information from the entire sequence [9, 5, 4]. The output of the feature encoder is discretized to qt with a quantization module Z $\rightarrow$ Q to represent the targets in the self-supervised objective (§ 3.2). It also builds context representations over continuous speech representations and self-attention captures dependencies over the entire sequence of latent representations end-to-end.” the model is further improved with Hidden-Unit BERT (HuBERT) approach proposed by (Hsu et al., 2021) and XLS-R for cross lingual speech representation by (Babu et al., 2021).

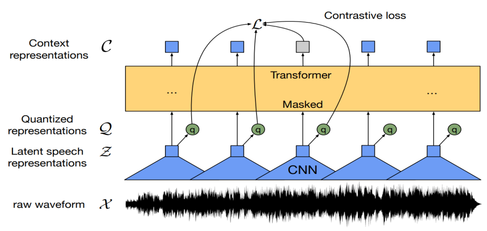


Figure 3: Wav2Vec architecture by Facebook AI

2.4 Language Model

A Language Model (LM) captures how words are typically used in a language to construct sentences or paragraphs. It could be a general-purpose model about a language such as English or Japanese, LM could be used to predict the next word in a sentence, to discern the sentiment of some text. Previously audio classification models required an additional LM and a dictionary to transform the sequence of classified audio frames to a coherent transcription based on context. Wav2Vec's architecture is based on transformer layers, thus having the context in consideration it can produce coherent transcriptions even without a language model. In addition, Wav2Vec2 leverages the CTC algorithm for fine-tuning, which solves the problem of alignment between a varying "input audio length"-to-"output text length" ratio. In conclusion, Wav2Vec model doesn't require a separate language model to generate acceptable transcripts. However, the performance of the model can be boosted by integrating a separate LM model to optimize the predicted transcript results even further. The language model should be good at modeling language that corresponds to the target transcriptions of the speech recognition system. In our experiments we have trained three language models via `kenlm` library Heafield (2011) i.e., 3-gram , 4-gram and 5-gram Language model and have used a 4-gram LM since it yields better scores in terms of WER. The model is trained to generalize the occurrence of maximum four consecutive words at a time also the model happens to correct a lot of word sequence errors of the Wav2Vec model transcription, as given in Table 1.

Table 1: Language Model Benchmarks

n-gram model	**WER**
3-gram	0.85122
4-gram	0.85095
5-gram	0.85097

2.5 Post-processing and Final Transcriptions

Post-Processing steps may consist of various methods like deep learning models or rule-based approaches such as Inverse Text Normalization (ITN) that is the task of converting the raw spoken output of the ASR model into its written form to improve text readability. Post-Processing is an important step to generalize transcriptions for readability and understanding. for e.g., spoken text is "I was born in nineteen eighty" or "my email is robert at the rate transformers dot com" in both the example transcriptions are not as generalized and would not look appealing in terms of readability, and in applications where a real-time speech to text system is been utilized such as Zoom meetings those kind of transcriptions won't be appreciated. So, once we apply ITN to the transcriptions the quality of the transcript readability increases significantly, let's take the previous examples and apply ITN to them, the output will be like - "I was born in 1980" and "my email is robert@transformers.com", here we can observe that the transcriptions are more generalized in terms of numbers and symbols. Similarly other post-processing approaches are such as Auto Punctuation , where we mostly use a deep Neural Network (DNN) to understand the context of the raw transcription and generate richer transcriptions with punctuation or capitalization based on context of the raw text. ASR systems typically generate texts without punctuation or capitalization, and punctuation can add an ability to understand the meaning of the text, where whole meaning of the sentence can change with a slight change in punctuation, that is the power this small post-processing step holds. Applying these post-processing methods produce even richer and readable transcriptions which makes the pipeline more robust and scalable.

3 Experiments and Comparative Analysis

This section presents the experimental evaluations of the optimized approach, using Wav2Vec model with optimization methods mentioned in the former sections. For the experiments, we have used Docker Containers to reflect the constraints of mobile devices and the experiments are performed with 2 most common mobile configurations i.e., 2 cores CPU with 8 gigabytes of RAM and 2 cores CPU with 4 gigabytes of RAM. It has additionally been tested with a GPU configuration comprised of Tesla P100 for a better intuition of the performance. We have done a cross-comparative analysis of the system proposed with general wav2vec model inference as well as other state-of-the-art models like Quartznet by (Kriman et al., 2020) and Citrinet model by (Majumdar et al., 2021).

For the experiments, we used librispeech's test set and pre-processed it further to generalize it for metric calculations. Table 3, presents the benchmarking runs performed on different resource configurations and it can be observed that even though on a RAM restricted environment such as 4GB RAM environment with 2 core CPU our system did not crash and gave almost similar inference to the environment with 8GB of RAM.

Table 2: A summary of the experimental settings

Environment	**Configurations**
Machine/CPU	Intel Haswell
GPU	Tesla P100
Operating system	Ubuntu 20.04 LTS
Docker Image	Python Slim 3.9
Memory Config (RAM)	8 GB/4GB
CPU Config (Cores)	2
Neural network library	`PyTorch`

Table 3: Benchmarks on different resource configurations

Resource Config	**WER**	**Inference (s)**
2 CPU, 8GB RAM	0.85	6381
2 CPU, 4GB RAM	0.85	6456
1 GPU, 8GB RAM	0.85	356

Along with WER we have calculated Jaccard Similarity scores of predicted transcripts which is another metric for text similarity and the pipeline tends to achieve a jaccard score of 0.92 on the same librispeech test set. For the comparative analysis we have compared the results of our approach with different state of the art methods. Table 4, presents the comparative tests based on librispeech clean test set between different state-of-the-art models, it can be observed that the proposed method Word Error Rate is significantly lower when compared to other common ASR models and with lower WER we get more accurate predictions. A detailed comparison of other non auto-regressive models are presented in (Ng et al., 2021).

Table 4: Librispeech test set benchmarks on different models

Model	**WER**
Proposed Work	**0.85**
Nvidia Quartznet	3.78
Wav2Vec Large	1.8
Nvidia Citrinet	2.7

4 Conclusion

This paper presents, an optimized way of scaling offline ASR system to resource constrained environments, in our experiments we have noted a very significant improvement such as 0.85 of WER and pipeline inference on environment with 4 gigabytes of RAM and 2 CPU cores without facing any system crashes also the results on resource constrained environment are mostly similar to a high-end resource environment, some related work of speech recognition in low end systems is also presented by (Thomas et al., 2013) and (Bansal et al., 2018). The proposed flow showed a significant improvement in terms of scalability and performance and can be integrated and used in the advancement of the applications such as smart devices like voice assistant enabled speakers, offline voice typing in mobile devices, speech summarization and speech context moderation etc. However, there is always a room for improvement, some future research can still be performed on improving the inference of the proposed method in the constrained systems since faster inference with least resources is always the call of state-of-the-art methods. Exploring and researching to add novelties such as speech content moderation in ASR systems seems one of the promising directions of future work.

References

Arun Babu, Changhan Wang, Andros Tjandra, Kushal Lakhotia, Qiantong Xu, Naman Goyal, Kritika Singh, Patrick von Platen, Yatharth Saraf, Juan Pino, et al. 2021. Xls-r: Self-supervised cross-lingual speech representation learning at scale. *arXiv preprint arXiv:2111.09296*.

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460.

Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. 2018. Pre-training on high-resource speech recognition improves low-resource speech-to-text translation. *arXiv preprint arXiv:1809.01431*.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the sixth workshop on statistical machine translation*, pages 187–197.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3451–3460.

Samuel Kriman, Stanislav Beliaev, Boris Ginsburg, Jocelyn Huang, Oleksii Kuchaiev, Vitaly Lavrukhin, Ryan Leary, Jason Li, and Yang Zhang. 2020. Quartznet: Deep automatic speech recognition with 1d time-channel separable convolutions. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6124–6128. IEEE.

Somshubra Majumdar, Jagadeesh Balam, Oleksii Hrinchuk, Vitaly Lavrukhin, Vahid Noroozi, and Boris Ginsburg. 2021. Citrinet: Closing the gap between non-autoregressive and autoregressive end-to-end models for automatic speech recognition. *arXiv preprint arXiv:2104.01721*.

Anna Mauranen and Svetlana Vetchinnikova. 2017. Chunks in which we process speech. In *14th International Cognitive Linguistics Conference (ICLC), Tartu, July*, pages 10–14.

Edwin G Ng, Chung-Cheng Chiu, Yu Zhang, and William Chan. 2021. Pushing the limits of non-autoregressive speech recognition. *arXiv preprint arXiv:2104.03416*.

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. IEEE.

Samuel Thomas, Michael L Seltzer, Kenneth Church, and Hynek Hermansky. 2013. Deep neural network features and semi-supervised training for low resource speech recognition. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6704–6708. IEEE.

Knowledge Graph-based Thematic Similarity for Indian Legal Judgement Documents using Rhetorical Roles

Sheetal S, Veda N, Ramya Narasimha Prabhu, Pruthvi P and **Mamatha HR**
Department of Computer Science and Engineering, PES University, India
{sheetals410,vedan1408, ramrag0107, mail.pruthvip}@gmail.com
mamathahr@pes.edu

Abstract

Automation in the legal domain is promising to be vital to help solve the backlog that currently affects the Indian judiciary. For any system that is developed to aid such a task, it is imperative that it is informed by choices that legal professionals often take in the real world in order to achieve the same task while also ensuring that biases are eliminated. The task of legal case similarity is accomplished in this paper by extracting the thematic similarity of the documents based on their rhetorical roles. The similarity scores between the documents are calculated, keeping in mind the different amount of influence each of these rhetorical roles have in real life practices over determining the similarity between two documents. Knowledge graphs are used to capture this information in order to facilitate the use of this method for applications like information retrieval and recommendation systems.

1 Introduction

The Indian legal system is currently facing the problem of legal pendency owing to the large volume of cases that are filed each day. This is exacerbated by the lack of trained legal professionals and absence of good resources to aid legal experts. Thus, creating sound legal tech systems, especially ones aided with the advancements that have been seen by the field of artificial intelligence, is imperative for remedying the current situation.

Identifying similarity between documents isn't a new task, but identifying the similarity between legal text documents is challenging, understudied and quite essential. A mechanism that can identify the similarity of legal cases quite clearly would make a great backbone for a powerful information retrieval engine or a recommendation system that would greatly boost any legal expert's research and preparations.

Presently, popular legal information retrieval systems like `https://indiankanoon.org/` often use plain full-text similarity that doesn't make use of any kind of context, semantic or otherwise.

Through this work, the task of legal case similarity is explored using knowledge graphs. To accomplish this, the use of semantic segments present in case documents, identified using deep learning, and the jurisdiction of the case itself are implemented. This information is then used to calculate similarity scores for case documents. A knowledge graph is used to store the extracted information because of its ability to capture the relationships between the documents. The intent behind creating such a system is to provide a sturdy and reliable backbone to information retrieval engines that can be used by legal experts and laymen alike for research, preparation, study, etc.

The paper is structured as follows: The subsection 2 illustrates the related work of researchers, section 3 shows the framework. Section 4 explains the methodology, that includes, description of the dataset, data-preparation steps, similarity metrics and methods used and details of how the case documents are stored in a knowledge graph. Section 5 discusses the result and application of this work in the real world. Finally, section 6, concludes the paper with a brief view about future scope.

2 Related Work

2.1 Rhetorical Role Identification

The work in Bhattacharya et al. (2019) aims to use neural models or deep learning models for the "task of rhetorical role identification." There were some prior attempts made that relied on hand-crafted features which had a few disadvantages. It gave reliable results for a few domains only and it required legal knowledge which was expensive to get. Hence neural models were chosen as it does not rely on any hand-crafted features. The two neural models used for the task are the Hierarchical-BiLSTM model and Hierarchical-BiLSTM-CRF.

The results showed that the Hierarchical-BiLSTM-CRF model performed a little better than the other. The performance improvement was not so significant because CRF was unable to learn the emission score and transition score well. This happened because the legal document consists of a large number of sentences and only a few of them were considered for training purposes.

The paper Kalamkar et al. (2022) offers a corpus of English-language court judgment papers that are divided into relevant and cohesive sections. Each of these elements is labeled with a selection from a list of rhetorical roles. Based on the annotated corpus, they create baseline models for automatically predicting rhetorical roles in legal documents. There are 26,304 sentences annotated with 12 different rhetorical roles in the produced corpus, which comprises 265 Indian legal texts annotated with rhetorical roles. A transformer-based baseline approach for automatically annotating legal texts with sentence-level RR is also described in the paper. Finally, the research demonstrates how rhetorical roles can be used to improve legal summarization.

Majumder and Das (2020) worked on using models like Random Forest, Universal Sentence Encoder, BERT, and ROBERTA for labelling the rhetorical roles (RR). Sixty legal case documents from the Supreme Court were considered for this task. Fifty case documents were used for training and 10 for testing. Among all the models tried, ROBERTA outperformed the others. The output of ROBERTA was sent to BiLSTM. Three different models based on ROBERTA were tried each with different epochs. The first model was trained for 13 epochs, the second for 15, and the third trained for 19 epochs. Among the three models, the one trained for 15 epochs outperformed with a better Macro F-Score. However, the model was unable to label some RRs accurately.

2.2 Knowledge Graphs

The data used in Dong et al. (2021) is created as a result of extraction from semi-structured web pages and the attributes and relationships from the text are extracted using a Bi-GRU model. The graph constructed is visualized for better understanding on the China judgment Online website. However, they do not experiment with case similarity itself as a feature.

Incorporating features relevant to the legal domain, this work Dhani et al. (2021) on case similarity and citation-linked prediction builds on the use of knowledge graphs in Natural Language Processing tasks. For unstructured text from a corpus of court cases, laws, and rulings in Indian courts, a legal knowledge graph is constructed to represent the entities of a document and the relationships between them. Latent Dirichlet Allocation (LDA) is applied to model the most relevant topics for the derived ontology. Graph neural network models are used to identify missing links in a case graph constructed, with citation and similarity as relations, that has keywords/phrases specific to legal practice as node attributes. The performance of the relational graph convolutional networks on both tasks is shown to be higher when trained on the feature set containing lawpoints.

The paper Zhao et al. (2022) on legal judgment prediction deals with determining the law article, charge, and term of penalty given a fact description using graph neural networks. Judicial document text from the CAIL2018 dataset represented as 6 kinds of graphs which include co-occurrence, pointwise mutual information, semantic (using cosine similarity) and distance-based (Euclidean, Manhattan, and Chebyshev) underwent information updation between the graphs using a graph convolutional network. The prediction model fed with a fusion of information from graph nodes and law articles surpasses the baseline models in accuracy, macro-precision, macro-recall, and macro-F1 scores. The proposed method has an edge owing to the legal text differentiation extractor based on graph attention networks.

Usage of knowledge graphs for legal tasks has been undertaken before in Cavar et al. (2018), Filtz (2017) and Mandal et al. (2017) but these approaches don't make use of semantic segments which provide essential context for identifying similarity. Law practitioners often use their knowledge of the semantic segments in legal cases implicitly to identify similar cases. This, therefore, provides a strong motivation for identifying rhetorical roles and using knowledge graphs to capture them.

Works like Dhani et al. (2021) also make use of metadata like judge, court, and date. Approaches such as these end up relying less on law and more on context provided by such metadata which introduces bias and makes for a technology that isn't developed with fairness and equity in mind.

Use of knowledge graphs as the preferred method of storing information is motivated by the fact that it opens up the avenue to develop faster information systems and recommendation systems using the same strategy as the basis.

2.3 Case Similarity

Mandal et al. (2017) aims to improvise the text-based methods that are used to compute the similarities between documents and tried topic modelling, word, and document embedding. Among the different approaches tried, embedding based methods outperform the others. These approaches were tried on different representations of the documents like whole document, paragraphs, summaries, and text around citations. With summaries topic modelling (LDA) performed the best and with the whole document as a method of representation, Doc2Vec outperformed the other methods considered. However, there were a few drawbacks with summaries as a method of representation used because the results depended on the quality of the summaries and not all summaries were of high quality.

In the paper Bithel and Malagi (2021), unsupervised algorithms are used to rank documents based on their similarity to the query and top x documents are termed relevant documents. Approaches used by them include TFIDF with cosine similarity, Word embeddings, best match 25, TFIDF and BM25, Rake + TF-IDF, and cosine similarity sentence BERT. However, they do not use any of the more complex deep learning-based approaches and the possibility of training a BERT model from scratch so that the semantic meaning of prior cases and queries can be used.

The work Ostendorff et al. (2021) on legal literature recommendation aids research for a particular case by retrieving other decisions covering the same topic or necessary background information. Divided into (1) text-based (baseline TF-IDF, word-vector-based and transformer-based), (2) citation-based and (3) hybrid, 27 methods are compared on the basis of document length, and citation count and recommendation coverage over two datasets containing 2964 US case law documents. The results point to fastTextLegal as the overall best performing method. It is observed that the performance of text-based approaches such as Paragraph Vectors and Longformers is adversely affected by increasing word count and that of citation-based methods such as DeepWalk and Poincaré by decreasing citation count. Hybrid methods offer broad coverage and overcome the limitations of single methods.

In this paper Bhattacharya et al. (2020), similarity computation methods for legal documents based on textual content as well as precedent citation networks are analyzed on a common dataset of 47 pairs of Indian Supreme Court case documents. It describes text-based similarity measures like Paragraph Links, FullText Similarity using Doc2vec and proposes a novel method that considers aggregated scores between thematic segments (facts, arguments, rulings, statutes, etc) and network-based metrics like Bibliographic Coupling, Co-citation, Dispersion and Node2Vec, a unique algorithm for graph embeddings. Compared on the basis of their Pearson correlation coefficient, Node2Vec, Full-Text, and Thematic Similarity show comparable results. It is noted that a higher score is obtained with a combination of the 2 methods.

Previous works present several methods and approaches for similarity between legal documents, predominantly using full-text or citation-based techniques. The objective of our proposed solution is to cater to a particular user, here, law practitioners, by modeling their requirements while retrieving similar cases. Similarity is calculated using TF-IDF with cosine similarity and represented using the weighted average score between various rhetorical roles by effectively capturing their relative relevance.

3 Proposed Methodology

The intent with this work was to create a framework that is robust, reliable and focused on equity without bias . Therefore, the proposed solution introduces a framework that operates on an input case judgment document and generates as output the most similar case documents. The database consists of documents from the ILDC (Indian Legal Document Corpus) dataset. Rhetorical roles are then identified from all documents, similarity scores computed between the segments and represented as a knowledge graph as depicted in Figure 1.

4 Implementation

4.1 Dataset Used

The ILDC dataset, introduced in Malik et al. (2021), is used for this work . The dataset was originally created for the purpose of legal judgment prediction and explanation and contains judicial summary documents for more than 35,000 Supreme Court cases. The dataset is annotated with legal expert

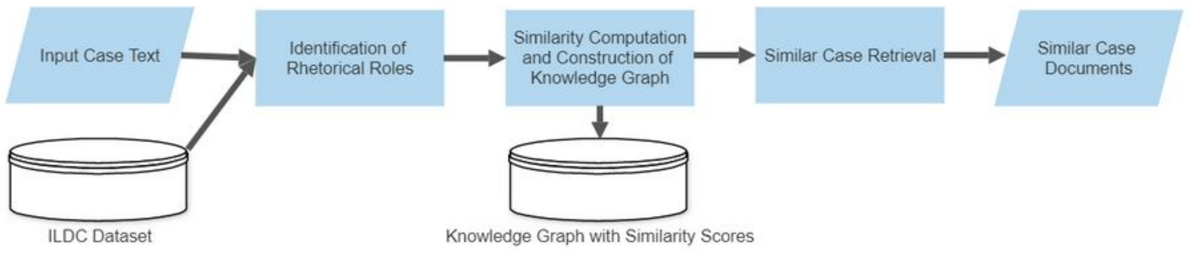

Input Case Text
Identification of Rhetorical Roles
Similarity Computation and Construction of Knowledge Graph
Similar Case Retrieval
Similar Case Documents
ILDC Dataset
Knowledge Graph with Similarity Scores

Figure 1: Diagrammatic Workflow of the Proposed Solution

guided explanations regarding the importance of different sentences in the judgment accompanied with the case. The motivation for using this dataset is the fact that it contains case documents without metadata like presiding judge and the place the case was filed which often introduce bias.

4.2 Identifying and labeling the relevant rhetorical roles

Rhetorical Role is also known as semantic segments. Each sentence in a legal case document can be assigned one of the predefined thirteen Rhetorical roles. Rhetorical role identification is a sentence classification task. The rhetorical role of a sentence does not only depend on the words in that sentence but also depends on the words from the sentence preceding and succeeding it. In other words, it depends on the context.

The impetus for identifying the rhetorical roles is based on the fact that the similarity between two cases is often ascertained based on the similarity between certain rhetorical roles.

To accomplish this task, the baseline method used by Kalamkar et al. (2022), which uses the SciBERT-HSLN architecture proposed by Brack et al. (2021) and is trained on the dataset contributed by the aforementioned work is adopted. Using this model, each line in the document is annotated with its corresponding rhetorical role as represented in Table 1.

4.3 Case Similarity

For a case document, once the rhetorical roles sentences are identified, all the sentences labeled with the same rhetorical role are grouped together. To determine how similar two case documents are, similarity scores between each of the thirteen rhetorical roles computed for both documents are being considered.

	Rhetorical Role	Label
1	Preamble	PREAMBLE
2	Facts	FAC
3	Ruling by Lower Court	RLC
4	Issues	ISSUE
5	Argument by Petitioner	ARG_PETITIONER
6	Analysis	ANALYSIS
7	Argument by Respondent	ARG_RESPONDENT
8	Statute	STA
9	Precedent Relied	PRE_RELIED
10	Precedent Not Relied	PRE_NOT_RELIED
11	Ratio of the decision	RATIO
12	Ruling by Present Court	RPC
13	None	NONE

Table 1: List of the Rhetorical Roles and their corresponding Labels

Similarity metrics like Jaccard Similarity, Euclidean Similarity, and Cosine Similarity were initially considered. To compute the similarity between documents, a few frequently used text embedding methods/techniques and deep learning-based textual similarity techniques were implemented.

4.3.1 Text Embedding Methods

TF-IDF: Term frequency is the normalized term count. Besides TF, another thing considered is how common a word is among all the documents and this is taken care of by the IDF. TF-IDF works by assigning more weightage to rare words and lesser weightage to commonly occurring words and, frequency of a word in a document relative to its frequency in the entire corpus can be found out.

Doc2Vec: Doc2vec generates numerical representations for sentences, paragraphs and documents. It represents the document into a vector of size 20 and hence there is no need to consider the average of word vectors to create document vectors.

Method	Precision	Recall	Micro F1
Jaccard	0.41	0.41	0.41
TF-IDF + Cosine	0.71	0.71	**0.71**
TF-IDF + Euclidean	0.43	0.43	0.43
Doc2Vec + Cosine	0.38	0.38	0.38
Doc2Vec + Euclidean	0.43	0.43	0.43
BERT + Cosine	0.47	0.47	0.47
BERT + Euclidean	0.43	0.43	0.43

Table 2: Scores for Similarity Computation Methods

4.3.2 Deep Learning-based Textual Similarity

BERT: BERT makes use of transformers, an "attention mechanism" to learn the contextual/semantic relations between words in the document. Since the transformer encoder reads the entire sequence at once (unlike directional models), it learns the context of the word based on the words towards its left and right. The BERT-base model fine-tuned for the NLI dataset was used to learn embeddings of the word in the document.

Legal experts were asked to rank a subset of documents from the ILDC dataset on the basis of their similarity to each other in order to understand how relevant the implemented systems are to the real-world requirement from the perspective of information retrieval systems of this kind. On comparing this with the implemented systems, TF-IDF, a text-based embedding method and cosine similarity as the similarity metric yielded the highest micro F1 score of 0.71, as shown in Table 2, and was chosen to compute the similarity.

4.4 Knowledge Graph Representation

The information regarding the case documents is stored in a knowledge graph as modelled in Figure 2. Knowledge graphs are abstract data structures that capture both the characteristics of entities and the relationships between them. They are formulated by: $G = (V, E)$

Legal documents are often related to each other thematically. Capturing this is likely to yield results that are better suited for real-world application. This intuition drives the use of knowledge graphs in this paper.

In the constructed knowledge graph, case documents are represented as nodes and each node has 13 attributes associated with them and they represent the 13 rhetorical roles that were identified and tagged in the documents. The value taken up by each attribute is the corresponding sentences in order to increase the robustness of the data captured.

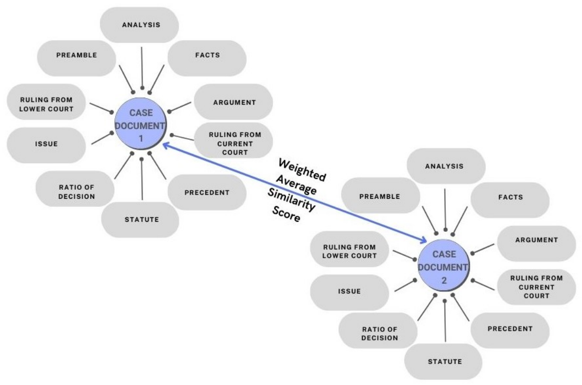


Figure 2: Knowledge Graph Representation

Rhetorical Role	Weight
ANALYSIS	5
FAC	
STA	4
RPC	
RATIO	
ISSUE	
RLC	3
NONE	
ARG_PETITIONER	2
ARG_RESPONDENT	
PRE_RELIED	
PREAMBLE	1

Table 3: Weights assigned to each Rhetorical Role

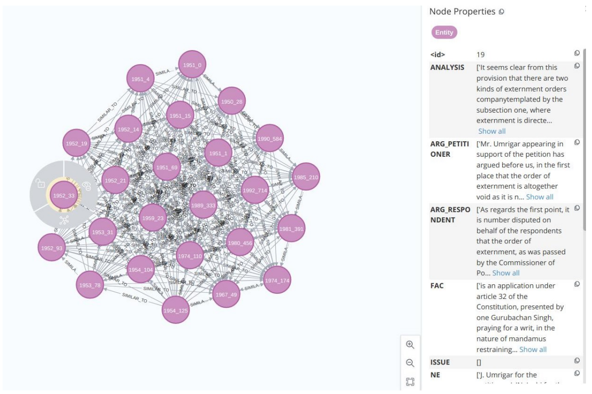
In practice, as pointed out by legal experts, different rhetorical roles have different amounts of relevance. To capture this as accurately as possible, the similarity scores of sentences in each individual rhetorical role in a case document are calculated using TF IDF. A weighted average of the same is then taken to obtain the corresponding edge weight.

The weights were decided by conferring with legal experts, as shown in Table 3. The importance of each rhetorical role while gauging the similarity between case descriptions as well as the accuracy of the baseline model used to identify them were considered in order to reduce the value of error.

The final weighted average similarity score is calculated as:
$WeightedAverageSimilarityScore =$

$$\frac{\sum(Weights \times SimilarityScoreofRhetoricalRole)}{\sum Weights}$$

The weighted average similarity computation as per the priority order of the rhetorical roles resulted in a micro F1 score of 0.73, showing an improvement over the full-text methods.
The data structure containing the source and destination cases along with the similarity scores corresponding to the rhetorical roles thus obtained is stored both in a CSV and Neo4j, a graph database. The use of Neo4j as a database is dictated by its efficiency, ability to scale and ease of use. In order to ensure quick search and accurate retrieval, edges with low scores are dropped from the database. A snippet of nodes and their properties in the Neo4j database can be seen in Figure 3.


Node Properties

Entity

<id>	19
ANALYSIS	['It seems clear from this provision that there are two kinds of externment orders companytemplated by the subsection one, where externment is directe... Show all
ARG_PETITIONER	['Mr. Umrigar appearing in support of the petition has argued before us, in the first place that the order of externment is altogether void as it is n... Show all
ARG_RESPONDENT	['As regards the first point, it is number disputed on behalf of the respondents that the order of externment, as was passed by the Commissioner of Po... Show all
FAC	['is an application under article 32 of the Constitution, presented by one Gurubachan Singh, praying for a writ, in the nature of mandamus restraining... Show all
ISSUE	[]
NE	['J. Umrigar for the

Figure 3: Representation of Knowledge Graph using Neo4j

5 Results

On comparing expert-anointed rankings for case documents based on their full-text similarity, TF-IDF with cosine similarity metric was observed to capture it most effectively with the highest micro F1 score of 0.71, amongst other methods such as BERT and Doc2Vec. Rhetorical roles were identified using the SciBERT-HSLN approach which achieved a micro f1 of 77.7 on hidden test data. Similarity scores were further enhanced to achieve a micro F1 score of 0.73, by integrating the aspect of thematic similarity using rhetorical roles. It was found to model expert-scores more accurately than the full-text methods alone. This information when represented in the form of a knowledge graph through the Neo4j database yielded an average query response time of 223 ms, showing an improvement nearly 10 times faster than retrieving the documents saved in a table. In a use case such as ours, it is of essence to note that the relationship between different documents is important. Only capturing course-grained details like document text and leaving similarity calculations to application logic leads to added computational costs incurred at every query, which can be avoided if this information is stored in a knowledge graph at the time that a new document is added to the database.

5.1 Applications in the Real World

5.1.1 Information Retrieval Systems

One of the most well-known applications of case similarity is information retrieval. Non-proprietary legal information retrieval systems like Indian Kanoon (https://indiankanoon.org/) have shortcomings which include the upper limit on the number of tokens that can be matched and the lack of robustness when the search is carried out.

An IR system backed by a database that stores documents like the one proposed in this paper can easily service multi-sentence queries by identifying the rhetorical roles present in the input and finding the corresponding weighted similarity scores.

This system can be further scaled over a large database with the help of clustering to reduce the number of similarity scores that need to be calculated.

5.1.2 Recommendation Systems

Recommendation systems can be used to suggest similar documents to law experts who are researching to make their preparation better. The usage of knowledge graphs makes it very easy to identify relevant similar documents faster.

The length of the feature vector for an item based collaborative filtering based recommender system with n users, m items, and c ratings is n; thus, the time complexity of the similarity computation is O (n). As a result, the overall temporal complexity is found to be $O(m^2n^2)$.

With the approach proposed by this work, the similarity computation has a time complexity of $O(|N|^2)$ (N being the number of nodes). The time complexity of the resulting recommendation system becomes $O(n * |N|^2)$.

6 Conclusion

This paper introduces a new method to compute and store similarity between judicial case documents. It identifies the rhetorical roles in case documents and leverages it to find the thematic similarity between the documents. The similarity scores thus obtained are stored in a knowledge graph along with the documents. In this representation the documents and their semantic segments are captured in the nodes and the similarity scores are represented as the edge weights. It is shown that this method can be used for building reliable information retrieval systems and recommendation systems.

For future work, using this data-structure for other legal tasks such as judgment prediction can be explored. It also remains to be determined if the task of explainability can be enhanced by the context provided by the identification of rhetorical roles. The weights in this work are determined using expert-rule methods. As an extension, different deep learning methods can be used to ascertain the optimal weights associated with the edges.

Acknowledgements

The authors thank Deeksha Kankane, a law student from Narsee Monjee Institute of Management Studies, Navi Mumbai who assisted this research by providing insights pertaining to the legal domain. The valuable technical and legal guidance provided by the professionals part of the team at Agami, India is gratefully acknowledged.

References

Paheli Bhattacharya, Kripabandhu Ghosh, Arindam Pal, and Saptarshi Ghosh. 2020. Methods for computing legal document similarity: A comparative study. *arXiv preprint arXiv:2004.12307*.

Paheli Bhattacharya, Shounak Paul, Kripabandhu Ghosh, Saptarshi Ghosh, and Adam Wyner. 2019. Identification of Rhetorical Roles of Sentences in Indian Legal Judgments.

Shivangi Bithel and Sumitra S. Malagi. 2021. Unsupervised Identification of Relevant Prior Cases. *CoRR*, abs/2107.08973.

Arthur Brack, Anett Hoppe, Pascal Buschermöhle, and Ralph Ewerth. 2021. Sequential Sentence Classification in Research Papers using Cross-domain Multi-task Learning. *CoRR*, abs/2102.06008.

Damir Cavar, Jon Herring, and Alison Meyer. 2018. Law Analysis using Deep NLP and Knowledge Graphs. In *Proceedings of the LREC 2018 "Workshop on Language Resources and Technologies for the Legal Knowledge Graph"*.

Jaspreet Singh Dhani, Ruchika Bhatt, Balaji Ganesan, Parikshet Sirohi, and Vasudha Bhatnagar. 2021. Similar Cases Recommendation using Legal Knowledge Graphs. *CoRR*, abs/2107.04771.

Biao Dong, Haoze Yu, and Haisheng Li. 2021. A Knowledge Graph Construction Approach for Legal Domain.

Erwin Filtz. 2017. Building and Processing a Knowledge-graph for Legal Data. In *The Semantic Web*, pages 184–194, Cham. Springer International Publishing.

Prathamesh Kalamkar, Aman Tiwari, Astha Agarwal, Saurabh Karn, Smita Gupta, Vivek Raghavan, and Ashutosh Modi. 2022. Corpus for Automatic Structuring of Legal Documents.

Soumayan Bandhu Majumder and Dipankar Das. 2020. Rhetorical Role Labelling for Legal Judgements Using ROBERTA. In *FIRE*.

Vijit Malik, Rishabh Sanjay, Shubham Kumar Nigam, Kripabandhu Ghosh, Shouvik Kumar Guha, Arnab Bhattacharya, and Ashutosh Modi. 2021. ILDC for CJPE: Indian legal documents corpus for court judgment prediction and explanation. 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 4046–4062, Online. Association for Computational Linguistics.

Arpan Mandal, Raktim Chaki, Sarbajit Saha, Kripabandhu Ghosh, Arindam Pal, and Saptarshi Ghosh. 2017. Measuring Similarity among Legal Court Case Documents. In *Proceedings of the 10th Annual ACM India Compute Conference*, Compute '17, page 1–9, New York, NY, USA. Association for Computing Machinery.

Malte Ostendorff, Elliott Ash, Terry Ruas, Bela Gipp, Julian Moreno-Schneider, and Georg Rehm. 2021. Evaluating document representations for content-based legal literature recommendations. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law*, pages 109–118.

Heng-Ru Zhang, Fan Min, Zhi-Heng Zhang, and Song Wang. 2019. Efficient collaborative filtering recommendations with multi-channel feature vectors. *International Journal of Machine Learning and Cybernetics*, 10.

Qihui Zhao, Tianhan Gao, Song Zhou, Dapeng Li, and Yingyou Wen. 2022. Legal Judgment Prediction via Heterogeneous Graphs and Knowledge of Law Articles. *Applied Sciences*, 12:2531.

SConE: Contextual Relevance based Significant CompoNent Extraction from Contracts

Hiranmai Sri Adibhatla, Manish Shrivastava
Language Technologies Research Center, KCIS
IIIT Hyderabad, India
hiranmai.sri@research.iiit.ac.in,m.shrivastava@iiit.ac.in

Abstract

Automatic extraction of “significant” components of a legal contract, has the potential to simplify the end user’s comprehension. In essence, “significant” pieces of information have 1) information pertaining to material/practical details about a specific contract and 2) information that is novel or comes as a “surprise” for a specific type of contract. It indicates that significance of a component may be defined at an individual contract level and at a contract-type level. A component, sentence or paragraph, may be considered significant at a contract level if it contains contract specific information (CSI), like names, dates or currency terms. At a contract-type level, components which deviate significantly from the norm for the type may be considered significant (type specific information (TSI)). In this paper, we present approaches to extract “significant” components from a contract at both these levels. We attempt to do this by identifying patterns in a pool of documents of the same kind. Consequently, in our approach, the solution is formulated in two parts: *identifying CSI* using a BERT based contract-specific information extractor and *identifying TSI* by scoring sentences in a contract for their likelihood. In this paper, we even describe the annotated corpus of contract documents that we created as a first step toward the development of such a language-processing system. We also release a dataset of contract samples containing sentences belonging to CSI and TSI.

1 Introduction

Contracts are agreements, between two or more parties, that govern what each party can or cannot do and are usually dense in information. Extracting contract elements and locating novel clauses and assignments from a legal contract is a desired feature by many as it will greatly simplify and accelerate user comprehension. Traditionally, it requires a domain expert as there are parts of a contract that can only be noticed by a reader experienced in reviewing contracts. For an untrained eye, it is often difficult and time consuming to identify rare and unique sentences. To reduce dependency on experts and to lessen the human effort required, in this paper we introduce approaches for automatic identification and extraction of significant components of the contract.

When compared with the corpora on which most pre-trained deep models are based, the structure and vocabulary of texts in contracts differ significantly. Contracts frequently take constrained forms, sometimes even “template-like” for the sake of ensuring legal unambiguity. On carefully examining the semantics and structure of diverse legal contracts sourced from SEC EDGAR [1] *(employment, software license, purchase, severance)*, we observe that

i) *within contracts of same category*, although the wording and sentence structure differ between individual contracts, the information conveyed remains almost the same,

ii) *within an individual contract*, within an individual contract, we have compo- ii) within an individual contract, we have components, sentences or paragraphs that are remarkably distinct with little redundancy

Components in an individual contract can be broadly classified as:

Templatised sentences are sentences that follow a template. A phrase or only a part of the sentence may vary and the rest of the content is semantically same across contracts. Examples include contract elements (Chalkidis et al., 2017) like title of the contract, parties involved in the contract, dates, governing law.

As observed in Table 1, the sentences for an individual contract can be generated from a template by filling in relevant information for the "effective date" and "governing law". The values for "effec-

[1]https://www.sec.gov/edgar/search-and-access

Templatised Sentences
This Agreement shall be effective as of *November 5, 2014 (the Effective Date).*
GOVERNING LAW.This Agreement shall be construed and interpreted in accordance with the internal laws of the *State of California*.

Table 1: Sentences with a Template Structure

Boilerplate Sentences
While employed by the Company hereunder, Executive shall be eligible to participate in the Company 's employee benefit plans as in effect from time to time pursuant to the terms of those employee benefit plans.
No waiver of any breach or condition of this Award Agreement shall be deemed to be a waiver of any other or subsequent breach or condition whether of like or different nature.

Table 2: Sentences with Standardized Clauses

Rare Sentences
To achieve balance, your current tax withholdings may cease and a hypothetical rate of tax may be calculated and withheld from your wages.
No additional Stock Units granted as part of the Award may be earned following the Change in Control.

Table 3: Sentences with Rare elements

tive date" and "governing law" will be different for different contracts. In templatised sentences, the information changes rapidly for each document, as the values are unique to each contract.

Boilerplate sentences[2] sentences that are standard formulations, and uniformly found in all contracts of a type. They are huge in number and constitute a large portion of contract. In this paper, we extend the definition of Boilerplate sentences to include sentences which contain the same semantic content across contracts but differ lexically and structurally. As we can see in Table 2, these clauses are standard across contracts of a specific type. Business and technical documents often use boilerplate sentences to improve efficiency and standardize language and structure. The information divergence between contracts of a type is almost constant for boilerplate sentences.

Rare sentences in a contract include content not commonly found in contracts of that type and hence are conspicuous by their presence in the current contract. In Table 3, first example refers to hypothetical tax rate which applies to employees who work at an onsite location. Similarly in the second example, no additional stock units are granted if there is a change in control of the organization. Both these clauses are situational and do not appear in most contracts of that category. Intuitively, these sentences will be of interest to anyone examining the contract because they bring in novelty. Rare sentences in a contract are identified on the basis of contract type to which a contract belongs.

In summary, a contract has -

i) template sentences, which contain contract specific information (CSI) and are generic across contract types.

ii) rare sentences which deviate from other contracts of the same type, they convey type specific information (TSI) and can be recognised only if one has an in depth understanding of the content usually present in the contract type.

iii) common and well understood clauses that constitute boilerplate sentences. In terms of volume, they account for majority of the sentences in a contract.

This approach of extracting significant components does not really qualify as a standard summarization task because there is no merit in summarizing boilerplate sentences which are well understood. Abstractive summarization (Zhang et al., 2020a) techniques would inadvertently change the semantics of the contract. Even when compared to an extractive setting (Nallapati et al., 2017), in this study our main focus is to accentuate rare and templatised sentences as significant components in comparison to boilerplate sentences. Unfortunately, we are not aware of any large, open corpora of contracts for running comparable experiments.

The outcome of our approach is presented in two formats *a) highlighted input document* Figure 1- where sections of interest are highlighted within the overall contract. This helps in vizualizing the significant components of the contract. *b) a cover-page* - a consolidated page containing the extracted significant components. The effectiveness of the automated significant components identification model was further evaluated by conducting an experimental study that compares the performance between human and machine for the task. The contribution of our work in addition to identifying “significant" components is to understand

[2]The term boilerplate refers to standardized text, copy, documents, methods, or procedures that may be used over again without making major changes to the original.

how much fine-tuned data is required for achieving a moderately reasonable accuracy. This becomes important as contract types can vary considerably, and organizations would be burdened with huge annotation efforts for every document type.

2 Related Work

Language being the core of law and legal contracts, an increased interest in applying natural language processing techniques to a wide range of problems ranging from information extraction to sentence prediction in law (Zhong et al., 2020; Hendrycks et al., 2021; Kalamkar et al., 2021; Zheng et al., 2021) has been observed. Considerable amount of work has been done in contract analysis and information extraction from contracts (Yang et al., 2013; Silva et al., 2020; Mittal et al., 2015).

The most obvious approach to automatic contract element extraction is to model it as sequence labeling task. Statistical methods like Conditional Random Fields (Finkel et al., 2005; Xu and Sarikaya, 2013) were popular for sequence labeling prior to neural networks. (Chalkidis et al., 2017) involved hand written rules along with hand crafted features to uniquely identify and extract the contract elements. Recently, neural networks (Huang and Xu, 2015; Ma and Hovy, 2016; Chalkidis and Androutsopoulos, 2017) and BERT (Devlin et al., 2019) based approaches (Zhang et al., 2020b; Chen et al., 2019) were developed for sequence labeling and slot with joint intent classification. Our work for CSI extraction closely resembles (Zhang et al., 2020b) where the contract elements are extracted from regulatory filings and property lease agreements using the standard BIO tagging scheme for the contract elements of interest. We include more categories of contracts (employment, incentive, purchase, severance, software-license) and the contract elements are majorly kept consistent.

Scope identification is another popular area of research in legal domain as it is tedious to read legal documents. Contracts or legal documents contain many key sentences. It often becomes necessary to have domain knowledge regarding contracts to avoid missing any important or key information. Summarization (Andhale and Bewoor, 2016) is a reliable approach and summarizing legal contracts was attempted (Kubeka and Ade-Ibijola; Kore et al., 2020) by taking the document features and ordering the sentences according to their importance. Classification and hand crafted rules (Le et al., 2020) was another recent approach to precisely identify the scope and was applied to construction contracts to identify requirements automatically. These techniques do not differentiate between boilerplate sentences which forms the bulk of the contract and the other sentences of the contract. Instead of summarizing well understood and accepted clauses, our study intends to focus on contract specific information (CSI) and contract type specific information (TSI).

Regression (Ren et al., 2016; Zopf et al., 2018) is another technique where the sentences are scored on their importance and the model learns to include sentences in a summary based on the scores it predicts. Based on our observations that legal contracts of same category have repetitive information, we devised an approach to calculate sentence likelihood with respect to the contract type and use these scores to identify TSI. The likelihood scores calculated using LaBSE (Feng et al., 2022) while BERT (Devlin et al., 2019) was adapted to learn and predict these likelihood scores.

3 Approach

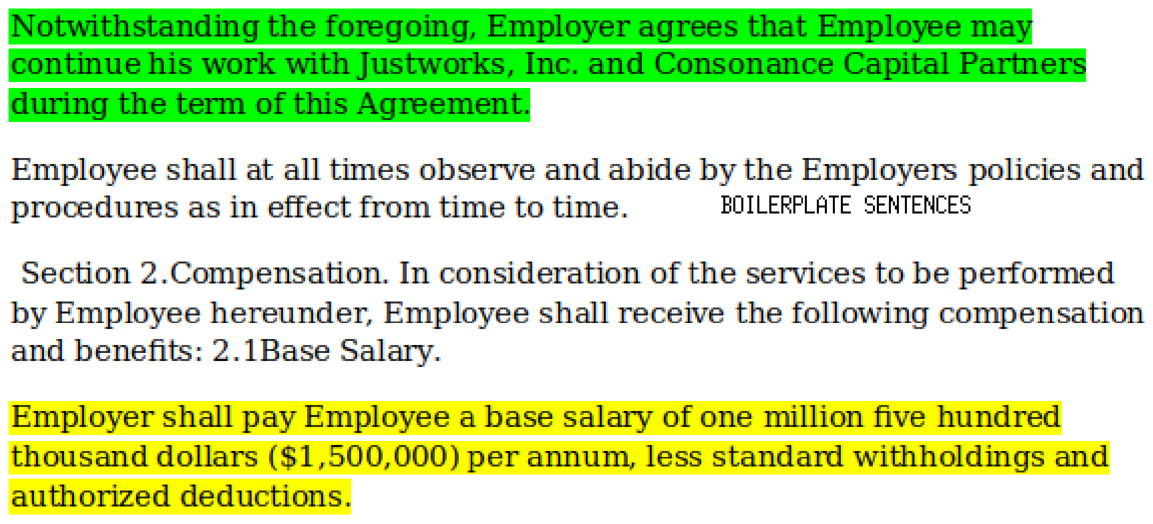
Significant component extraction is accomplished in two stages:

(1) Identifying CSI by processing each sentence of the document and identifying sentences with contract elements (Chalkidis et al., 2017).

(2) Identifying TSI by assigning a likelihood score to all sentences in a contract.

These stages contribute in effectively identifying the scope of significant components, by automating contract processing and extracting text relating to CSI and TSI from the contracts. We use LEGAL-BERT-BASE (Chalkidis et al., 2020) which is fine-tuned on BERT (Devlin et al., 2019) for legal domain and has shown substantial improvement in challenging downstream tasks like multi-label-classification. Within the wide categories of legal contracts available, we ran our experiments on the contract types mentioned in Table 9.

The overall architecture is shown in Figure 2. The input to the model is a document D containing a set of sentences S. The output is a set of sentences P, that effectively highlight information unique and specific to the document D, such that $P \in S$.



Notwithstanding the foregoing, Employer agrees that Employee may continue his work with Justworks, Inc. and Consonance Capital Partners during the term of this Agreement.

Employee shall at all times observe and abide by the Employers policies and procedures as in effect from time to time.

BOILERPLATE SENTENCES

Section 2.Compensation. In consideration of the services to be performed by Employee hereunder, Employee shall receive the following compensation and benefits: 2.1Base Salary.

Employer shall pay Employee a base salary of one million five hundred thousand dollars ($1,500,000) per annum, less standard withholdings and authorized deductions.

Figure 1: Example snippet of a highlighted contract. Sentence in green is TSI while the yellow sentence is CSI. The other two sentences are boilerplate.

4 Identifying CSI

Identifying contract elements is similar in approach to identifying named entities but is not directly extendable without retraining them on contracts. NER systems typically identify persons, organizations, dates, locations, currency terms etc,. Contract elements would carry more features attributed to it along with being a named entity. For example, a NER system can identify dates and persons but will not be able to differentiate if the date is an effective start date or termination date. Similarly not all instances of persons, organization or location in a contract would be contract parties or governing law elements. The sentences that contain these CSI are almost in a template like schema, therefore training a sequence labeling model to understand the sentence semantics and to extract sentences which contain contract elements, yields better results. We sampled 500 legal documents (100 documents of each category mentioned in Table 9). These documents are then pre-processed into paragraphs. A paragraph as a unit might be of a higher value than an isolated sentence. The documents are split into train, test and validation bins in the ratio 7:2:1. Commonly applicable contract elements are identified and selected as contract elements of interest. Most of the contract elements are phrases rather than a single token, therefore we pose it as a sequence labeling task using a standard BIO tagging scheme (Tjong Kim Sang, 2002). We manually annotated the contracts to mark the selected contract elements. The contract elements are kept consistent across the contract types as it is common for contracts to follow a fixed structure with a certain number of prescribed elements (*contract title, contract parties, effective start date, termination|maturity date, governing law etc.*). It also reduces the training and annotation effort and increases the generality of the model. The contract elements we annotated are listed in Table 7.

4.1 Identifying CSI Model

In the CSI model we extend BERT (LEGAL-BERT-BASE) for sequence labeling in order to identify phrases of interest. All contracts are divided into paragraphs. The input sequences are tokenized using BERT tokenizer and special tokens [CLS] and [SEP] are added at the beginning and end of the input sequence respectively. All the input sequences are padded to a maximum length of 256 tokens. After passing through BERT, we apply a linear layer and CRF layer on top of the hidden states output of the last layer. The model is trained for 25 epochs with learning rate of 1e-05.

5 Identifying TSI

Contract type specific information (TSI) extraction problem has not been studied extensively and is the main focus of our study. We identify unique or novel details concerning the contract by looking at structural and semantic similarities among a pool of contracts belonging to a specific type. A clause that is rare for an employment type contract may not be rare for a stock options awards type contract.

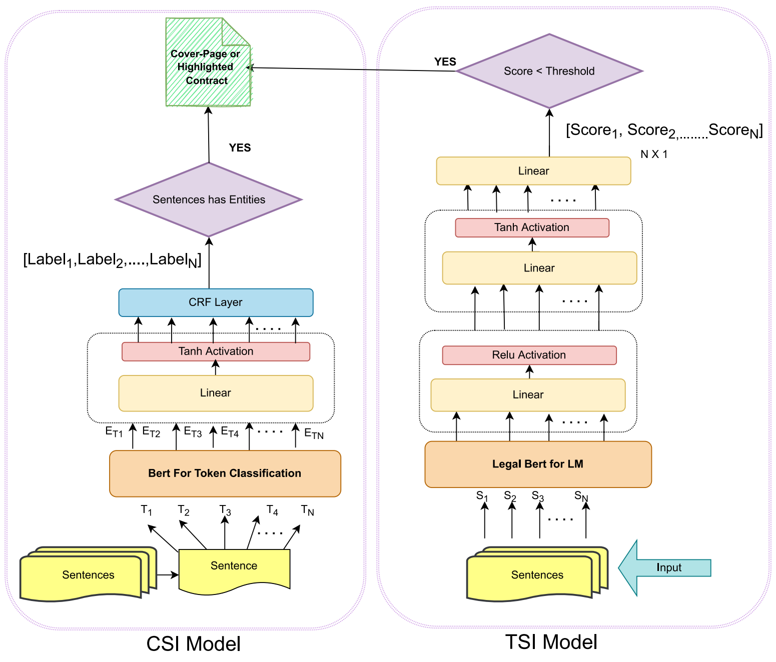


Figure 2: SCoNE Architecture for CSI and TSI

Figure 3 highlights few clauses that may seem ordinary but are different from their usual construction in contracts.

Based on our observations, legal contracts of same category have repetitive information (Boilerplate). This requires ranking the sentences based on a metric for rarity. Scoring sentences (Ren et al., 2016; Zopf et al., 2018) based on both importance and redundancy among sentences was attempted for summarization (Nallapati et al., 2017) tasks. The approach however, does not guarantee inclusion of rare and unique sentences as sentences scored based on their importance are most likely to pick boilerplate sentences since they are the core of any contract. Redundancy is almost negligible for business documents like contracts. TextRank (Mihalcea and Tarau, 2004) is a popular graph-based unsupervised ranking model for text processing. It identifies text units that best define the task at hand and links them with sub units of text by identifying relations among them. The Local Outlier Factor (LOF) algorithm (Pedregosa et al., 2011) is an unsupervised anomaly detection method which computes deviation of a given data point with respect to its neighbors. We applied both TextRank and LOF algorithm on our sampled data as baselines. Since our aim is to capture the rare sentences, we sorted TextRank scores in ascending order and considered the top sentences as rare. Though the model works well in capturing rare information, deciding on the threshold or cut-off is often difficult as it would differ from contract to contract and contract type to contract type.

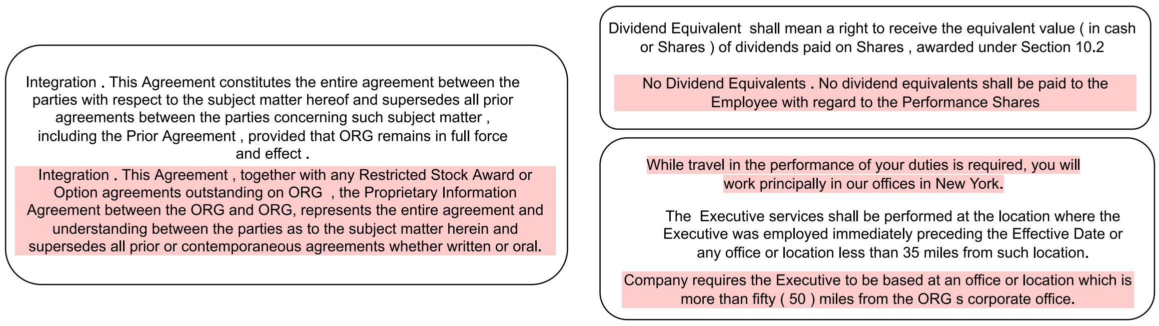
We devised an unsupervised approach to calculate TSI score with respect to the contract type and use these scores to identify TSI. TSI score of a sentence here indicates the confidence with which a given sentence is a part of a specific contract type.

Identifying rare components of a contract type is often limited by the presence of named entities in templatised sentences. These templatised sentences, though common across contract types, would be counted as rare by the virtue of having named entities in them. The information contained in such sentences is often extracted using Contract Element Extraction approaches (Chalkidis et al., 2017). In order to ignore these sentences and to make sentences more comparable across contracts we mask all the named-entities in contracts using spaCy[3] to replace named-entities by their type (people's names to PERSON, organization names to ORG). Masking sentences that contain named-entities increases its TSI score. Table 4 shows examples of few sentences whose TSI score has increased after masking named-entities.

5.1 Mean-Max Pooling

Though contracts of a type contain repetitive information, the vocabulary and structure might change.

[3]http://spacy.io



Integration . This Agreement constitutes the entire agreement between the parties with respect to the subject matter hereof and supersedes all prior agreements between the parties concerning such subject matter , including the Prior Agreement , provided that ORG remains in full force and effect .

Integration . This Agreement , together with any Restricted Stock Award or Option agreements outstanding on ORG , the Proprietary Information Agreement between the ORG and ORG, represents the entire agreement and understanding between the parties as to the subject matter herein and supersedes all prior or contemporaneous agreements whether written or oral.

Dividend Equivalent shall mean a right to receive the equivalent value (in cash or Shares) of dividends paid on Shares , awarded under Section 10.2

No Dividend Equivalents . No dividend equivalents shall be paid to the Employee with regard to the Performance Shares

While travel in the performance of your duties is required, you will work principally in our offices in New York.

The Executive services shall be performed at the location where the Executive was employed immediately preceding the Effective Date or any office or location less than 35 miles from such location.

Company requires the Executive to be based at an office or location which is more than fifty (50) miles from the ORG s corporate office.

Figure 3: Novel sentences snippets, highlighted in pink

Original Sentence	lscore	Entities Masked Sentence	lscore
EX-10.8 4 a17-1046_1 EX-10.8 EXHIBIT 10.8 EMPLOYMENT AGREEMENT This EMPLOYMENT AGREEMENT (the Agreement) is entered into and effective as of this 3rd day of March	0.75	EX-10.8 4 a17-1046_1 EX-10.8 EXHIBIT10.8 EMPLOYMENT AGREEMENT This EMPLOYMENT AGREEMENT (the Agreement) is entered into and effective as of this DATE DATE DATE (the Effective Date)	0.86
Term of this Agreement. The Term of this Agreement shall mean the period commencing on the Effective Date and ending on March 31	0.64	Term of this Agreement . The Term of this Agreement shall mean the period commencing on the Effective Date and ending on DATE DATE.	0.88

Table 4: Sentence Likelihood Scores (lscore) with and without Masking Entities

Textual overlap methods, therefore, would not be able to capture similar sentences across documents.

In order to estimate how frequently a sentence appears in the documents of a type, we compute semantic similarity between all sentences across all documents using LaBSE (Feng et al., 2022). We consider the maximum semantic overlap depicted by LaBSE as the indicator of semantic presence of the concept expressed by a sentence. Thus, we are approximating the expected count of a sentence (concept) occurring for a type using LaBSE score as proxy.

Let, S_{ij} be the j^{th} sentence in document D_{i} and S_{kl} be the l^{th} sentence in document D_{k}. Assuming no redundancy of concepts in legal contracts (each concept occurs once in a contract), we want to "count" the number of times a sentence appears in a document of a specific type. Thus,

$$Count_k(S_{\mathrm{ij}}) = \max_{1 \le l \le p} (LaBSE(S_{\mathrm{ij}}, S_{\mathrm{kl}})) \tag{1}$$

Where, p is the length of document D_k and LaBSE(S_{ij},S_{kl}) is the semantic overlap between the sentences. $Count_k(S_{\mathrm{ij}})$ would determine the degree of semantic overlap of S_{ij} with S_{kl}.

The "count" obtained for the sentence S_{ij} is mean pooled over the number of Documents N. This Mean-Max pooled LaBSE similarity score is assigned as the likelihood of a sentence.

The Likelihood score of S_{ij} is calculated using Equation 2.

$$TSIScore(S_{\mathrm{ij}}) = \frac{(\sum_{i=1}^{N} Count_k(S_{\mathrm{ij}})}{N} \tag{2}$$

Sentences that are very common across a type would have a higher likelihood score compared to sentences whose occurrence is semantically low. We are looking for sentences that have low mean similarity score i.e, low likelihood score.

5.2 Likelihood Approximation Model (TSI-A)

The process described in section 5.1 for calculating likelihood of each sentence would be quadratic in the total number of sentences and computationally expensive at runtime. Therefore, we train a BERT model on likelihood scores computed on contracts from five categories collected from SEC EDGAR. (Table 5) refers to contracts distribution in data slice of 5000 randomly selected files. This model would learn to predict the likelihood of a sentence given the contract type.

In the TSI-A model we extend BERT (LEGAL-BERT-BASE) for this regression. Documents are segmented into paragraphs and tokenized using

Contract Type	Number of Contracts
Employment	2200
Incentive	650
Severance	500
Purchase	750
Software License	600

Table 5: Contract distribution

BERT tokenizer, adding special tokens [CLS] and [SEP] at the beginning and end of the input sequence respectively. The input sequences are padded to a maximum length of 256 tokens. The final hidden states output is passed through linear layers with an activation layer in between for non-linearity. The last layer returns a score which serves as the sentence likelihood score. The model is trained for 15 epochs with learning rate of 1e-05. The loss criteria is MSE (Means Squared Error) and the objective is to minimise the loss between the predicted scores and the training scores. Pearson correlation scores are calculated between the test scores generated by using Equation 2 and the trained BERT regression model.

6 Human Evaluation

To assess the effectiveness of the TSI model we conducted an experimental study that compares the performance of the model against a human annotated corpus. Two annotators were asked to read the contract set provided to them and then label the sentences as rare or familiar. We chose to make this a binary classification task for the humans in order to reduce cognitive load.

Model generated scores in the test set were converted to labels based on their likelihood scores thresholded by the *knee-point* value for each class in Figure 4. If the sentence score is below the threshold set for rare sentences, then the sentences are labeled rare (0) . If the sentence score is above the threshold (set for rare sentences), then it is labeled familiar (1) . Table 6 details the precision, recall and f1 scores of both the annotators on selected contract types.

	Annotator 1			**Annotator 2**		
Contract Type	P	R	F_1	P	R	F_1
Employment	0.94	0.94	0.94	0.88	0.93	0.91
Incentive	0.92	0.97	0.94	0.92	0.90	0.91
Severance	0.99	0.90	0.94	0.99	0.83	0.90
Software License	0.99	0.94	0.96	0.99	0.87	0.93

Table 6: Human Evaluation Statistics

7 Evaluation

For evaluation, the masked contracts in the test set are divided into paragraphs, tokenized using BERT tokenizer and padded with special tokens ([CLS] and [SEP]).

7.1 CSI Model Evaluation

	F_1	*P*	*R*
ContractParties	0.92	0.89	0.95
ContractTitle	0.81	0.72	0.94
EffectiveDate	0.84	0.80	0.89
GoverningLaw	0.55	0.40	0.86
EmploymentRole	0.42	0.42	0.42
SalaryCompensation	0.49	0.43	0.57
TerminationDate	0.40	0.60	0.30

Table 7: Evaluation of Contract Elements

The table 7 shows micro-averaged metrics F_1, precision and recall across the selected contract elements. By examining these results, we can infer that common elements like ContractTitle, ContractParties, EffectiveDate which occur in all documents are well generalised by the BERT model and so have higher precision and recall values. The precision and recall scores are low for contract elements like TerminationDate, SalaryCompensation which have not commonly occurred in the test contracts sampled. The primary reason contributing to these low values is that contracts are sometimes amendments to pre existing contracts and they may not have all the contract elements that a new contract would mention. Table 8 shows the frequencies of the contract elements in both train and test bins after deduplication. The low representation of TerminationDate and SalaryCompensation samples in the train and test data explains low precision and accuracy values. The positives from this result is BERT is able to generalise commonly occurring contract elements with samples as low as 100 contracts. For uncommon contract elements, it requires more data.

7.2 Pearson Correlation Evaluation

Fig 4 shows the plots for sorted likelihood scores of sentences for each contract type. We observed that the plot is similar across contract types mentioned in Table 9 under contract types. From the plot we inferred that likelihood scores of sentences follow a trend. For all contract types, there exists sentences that have low likelihood and sentences which are more probable.

Contract Element	Frequency in Train Data	Frequency in Test Data
ContractParties	218	62
EmploymentRole	179	52
EffectiveDate	131	32
GoverningLaw	83	22
ContractTitlle	80	15
TerminationDate	38	3
SalaryCompensation	12	2

Table 8: Frequency of Contract Elements in Train and Test data

Contract Type	Pearson Correlation on Kfold
Employment	0.996
Incentive	0.998
Severance	0.990
Software License	0.997
Purchase	0.987

Table 9: Averaged K-Fold Validation for Pearson Correlation of test and predicted likelihood scores

i) lower likelihood score : these sentences map to rare sentences, not normally present in all the contracts of that category.

ii) average and above likelihood score : these sentences map to boilerplate sentences which uniformly occur in all the contracts with a minor change in wordings or expression and core sentences that contain named entities. Masking the named entities increases the likelihood scores of the templatised sentences.

Table 4 identifies few examples and compares original unmasked sentences with sentences masked using spaCy, where 'lscore' refers to the likelihood score. We observe that masking entities has shown impact on the sentence likelihood scores.

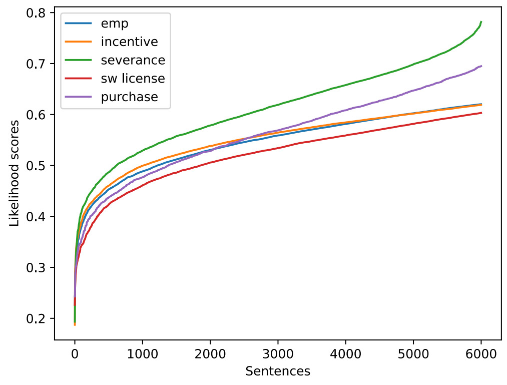


Figure 4: Sorted Likelihood Scores of Sentences

$$C = \frac{(N\sum_{i=1}^{N} T_i P_i - (\sum_{i=1}^{N} T_i)(\sum_{i=1}^{N} P_i))}{\sqrt{N\sum_{i=1}^{N} {T_i}^2 - (\sum_{i=1}^{N} T_i)^2}\sqrt{N\sum_{i=1}^{N} {P_i}^2 - (\sum_{i=1}^{N} P_i)^2}} \tag{3}$$

To measure the performance of our proposed model in predicting the likelihood score, we compute the Pearson product-moment correlation (C) (Benesty et al., 2009) between likelihood scores computed by mean pooling LaBSE similarity scores (calculated using Equation) (T) and likelihood scores generated by the TSI-A model (P), for a sample of 10000 sentences (N) using (Equation 3). Pearson correlation estimates the degree of statistical relationship between two independent variables. A high positive correlation between the actual and predicted values implies that the model can be trusted to work reasonably well on new unseen contracts of that category. For calculating the sentence Likelihood using TSI model, K-fold validation (with k=3) was performed. Table 9 has the Pearson correlation scores averaged for K-fold data sets on contract types considered. The high Pearson correlation values instill confidence that the model can identify rare sentences with reasonable accuracy.

7.3 TSI-A Model Evaluation

To the best of our knowledge, there are no publicly available corpora for rare sentence identification. But, rare sentence identification can be considered as either a ranking task or as an outlier detection task. Therefore, TSI-A model was evaluated against TextRank and LOF outlier detector applied on the sampled data. To keep the evaluation on similar grounds, we converted the likelihood scores obtained using TSI model to labels (0,1) based on the knee-point. On the 100 contracts sampled for test from each contract type, the contracts were split into train, test and validation bins in the ratio 7:2:1.

The performance of TextRank model was measured by considering the first 15 , 25 and 50 sentences as rare. Results were compared with human labeled data and Table 10 shows the precision, recall and f_1 values for all the thresholds considered. The metrics (precision, recall and f_1) were calculated for a document and then averaged for all the contracts. From the Table 10 we can observe that TextRank with threshold as 15 performs the best.

Although anomaly detection techniques are famous for identifying rare components, its applica-

	Text Rank								
	P_{15}	R_{15}	$F1_{15}$	P_{25}	R_{25}	$F1_{25}$	P_{50}	R_{50}	$F1_{50}$
Employment	0.83	0.90	**0.87**	0.84	0.85	0.85	0.84	0.74	0.79
Incentive	0.90	0.91	**0.91**	0.90	0.85	0.87	0.9	0.71	0.79
Severance	0.82	0.84	**0.83**	0.82	0.74	0.78	0.82	0.52	0.64
Software License	0.83	0.88	**0.86**	0.83	0.82	0.82	0.82	0.64	0.72
Purchase	0.90	0.88	**0.89**	0.90	0.8	0.85	0.89	0.61	0.72

Table 10: Evaluation of TextRankScores

	TSI			TextRank			Lof Outlier		
	P	R	F_1	P	R	F_1	P	R	F_1
Employment	0.89	0.93	**0.911**	0.83	0.90	0.87	0.92	0.44	0.59
Incentive	0.93	0.98	**0.96**	0.9	0.91	0.91	0.91	0.28	0.44
Severance	0.84	0.98	**0.91**	0.82	0.84	0.83	0.85	0.21	0.33
Software Licence	0.89	0.99	**0.93**	0.83	0.88	0.86	0.86	0.26	0.36
Purchase	0.92	0.97	**0.94**	0.9	0.88	0.89	0.92	0.44	0.59

Table 11: Evaluation of TSI, TextRank and LoF

tions on legal data are less prevalent. The main idea of unsupervised anomaly detection algorithms is to detect data instances in a dataset, which deviate from the norm. However, there are a variety of cases in practice where this basic assumption does not hold true. The anomalies could be local, global or anomalous when compared with its close-by neighborhood and determining a single approach that would work well for all data instances is difficult.

Table 11 compares the metrics of TSI model, TextRank and LoF outlier with the human labels. From the table it can be observed that TSI model performs better than unsupervised TextRank and LOF approaches.

The TSI model performs better at identifying the rare sentences than the best TextRank model as it is designed based on the semantics and structural features of legal contracts.

8 Conclusion and Future work

Our work is an attempt to study the structure of contracts and harness the semantic "strictness" of these contracts in order to extract "significant" pieces of information contained therein. Here, significance is defined by two distinct ideas: rarity in a type of contract and commonality across types. We find that this view of contracts removes a need to review elements which are boilerplate and would, in turn, reduce the effort required to find critical content in a given contract. We show that our models can achieve reasonable accuracy with relatively low training data. This work can be extended in future to a query based model by taking input from the users in the form of a query and highlight text most relevant to a given query. Since the task is novel and there exists no parallel corpora, we wish to release sentences, that are rare and sentences that contain contract specific elements from the sampled contracts.

9 Limitations

Our study aims at capturing significant components of a legal contract with an emphasis on identifying information that is specific and unique to a contract. While the approach successfully highlights and identifies significant components, there were a few limitations. The dataset contains contracts as well as amendments made to the existing contracts. These amendments contribute to low coverage of contract elements. Increasing the data for each contract type might yield in better coverage and results.

References

Narendra Andhale and LA Bewoor. 2016. An overview of text summarization techniques. In *2016 international conference on computing communication control and automation (ICCUBEA)*, pages 1–7. IEEE.

Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. 2009. *Pearson Correlation Coefficient*, pages 1–4. Springer Berlin Heidelberg, Berlin, Heidelberg.

Ilias Chalkidis and Ion Androutsopoulos. 2017. A deep learning approach to contract element extraction. In *JURIX*, pages 155–164.

Ilias Chalkidis, Ion Androutsopoulos, and Achilleas Michos. 2017. Extracting contract elements. In *Proceedings of the 16th edition of the International Conference on Articial Intelligence and Law*, pages 19–28.

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.

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*.

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.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2022. Language-agnostic BERT sentence embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891, Dublin, Ireland. Association for Computational Linguistics.

Jenny Rose Finkel, Trond Grenager, and Christopher D Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting of the association for computational linguistics (ACL'05)*, pages 363–370.

Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. 2021. Cuad: An expert-annotated nlp dataset for legal contract review. *arXiv preprint arXiv:2103.06268*.

Zhiheng Huang and Wei Xu. 2015. Kai yu. *Bidirectional LSTM-CRF models for sequence tagging. CoRR, abs/1508.01991*.

Prathamesh Kalamkar et al. 2021. Indian legal nlp benchmarks: A survey. *arXiv preprint arXiv:2107.06056*.

Rahul C Kore, Prachi Ray, Priyanka Lade, and Amit Nerurkar. 2020. Legal document summarization using nlp and ml techniques. *Int. J. Eng. Comput. Sci*, 9:25039–25046.

Sibusiso Kubeka and Abejide Ade-Ibijola. Automatic comprehension and summarisation of legal contracts. *contract*, 9:10.

Tuyen Le et al. 2020. Automated requirements identification from construction contract documents using natural language processing. *Journal of Legal Affairs and Dispute Resolution in Engineering and Construction*, 12(2):04520009.

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.

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, Barcelona, Spain. Association for Computational Linguistics.

Sudip Mittal, Karuna P Joshi, Claudia Pearce, and Anupam Joshi. 2015. Parallelizing natural language techniques for knowledge extraction from cloud service level agreements. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 2831–2833. IEEE.

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*.

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.

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, Osaka, Japan. The COLING 2016 Organizing Committee.

Paulo Silva, Carolina Gonçalves, Carolina Godinho, Nuno Antunes, and Marilia Curado. 2020. Using natural language processing to detect privacy violations in online contracts. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, pages 1305–1307.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *2013 ieee workshop on automatic speech recognition and understanding*, pages 78–83. IEEE.

Dan Yang, Christina Leber, Luis Tari, Aravind Chandramouli, Andrew Crapo, Richard Messmer, and Steven Gustafson. 2013. A natural language processing and semantic-based system for contract analysis. In *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, pages 707–712. IEEE.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020a. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Ruixue Zhang, Wei Yang, Luyun Lin, Zhengkai Tu, Yuqing Xie, Zihang Fu, Yuhao Xie, Luchen Tan, Kun Xiong, and Jimmy Lin. 2020b. Rapid adaptation of bert for information extraction on domain-specific business documents. *arXiv preprint arXiv:2002.01861*.

Lucia Zheng, Neel Guha, Brandon R Anderson, Peter Henderson, and Daniel E Ho. 2021. When does pretraining help? assessing self-supervised learning for law and the casehold dataset of 53,000+ legal holdings. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law*, pages 159–168.

Haoxi Zhong, Chaojun Xiao, Cunchao Tu, Tianyang Zhang, Zhiyuan Liu, and Maosong Sun. 2020. How does NLP benefit legal system: A summary of legal artificial intelligence. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5218–5230, Online. Association for Computational Linguistics.

Markus Zopf, Eneldo Loza Mencía, and Johannes Fürnkranz. 2018. Which scores to predict in sentence regression for text summarization? 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 1782–1791, New Orleans, Louisiana. Association for Computational Linguistics.

AniMOJity: Detecting Hate Comments in Indic languages and Analysing Bias against Content Creators

Rahul Khurana[*], **Chaitanya Pandey**[*], **Priyanshi Gupta**[*], **Preeti Nagrath**
Department of Computer Science and Engineering,
Bharati Vidyapeeth's College of Engineering,
New Delhi, India
{rahulkhurana.rk59, chaitanya.p2001, guppriyanshi, preetinagrath1}@gmail.com

Abstract

Online platforms have dramatically changed how people communicate with one another, resulting in a 467 million increase in the number of Indians actively exchanging and distributing social data. This caused an unexpected rise in harmful, racially, sexually, and religiously biased Internet content humans cannot control. As a result, there is an urgent need to research automated computational strategies for identifying hostile content in academic forums. This paper presents our learning pipeline and novel model, which classifies a multilingual text with a test f1-Score of 88.6 % on the Moj Multilingual Abusive Comment Identification dataset for hate speech detection in thirteen Indian regional languages. Our model, Animojity, incorporates transfer learning and SOTA pre- and post-processing techniques. We manually annotate 300 samples to investigate bias and provide insight into the hate towards creators.

1 Introduction

With the unbridled spread of Internet culture, hopes of finding an accepting community have led many racially, culturally, and sexually diverse groups to take refuge in their corner of the Internet, showcasing the strength of online forums. However, those seeking to spread hateful content look for ways to circumvent the restrictions placed on social media and hinder their healthy development. Due to the societal concern hate speech has garnered, there is a strong motivation to make advancements in its automatic detection. Online hate speech in general, and gendered online hate speech in particular, have become an issue of growing concern in both social and professional discourses.

Before we delve into detecting hate speech, it is imperative to understand its definition clearly. (Ross et al., 2017) believes that a distinct definition of hate speech can make the annotation process easier, leading to reliable detection of what categorizes as offensive. Nevertheless, hate speech and appropriate free expression walk a fine line, making its definition not fully agreeable. We opt to build upon existing definitions laid down by (Davidson et al., 2017), (De Gibert et al., 2018), and (Fortuna and Nunes, 2018) instead of proposing a specific definition.

Another issue that is not brought to light as often is that users with a diverse linguistic backgrounds tend to switch between different languages while expressing their thoughts on social media, limiting the capabilities of a monolingual model and necessitating the need for multilingualism in a model. We address this challenge by building a novel model that detects hate comments for thirteen regional languages (Hindi, Urdu, Telegu, Marathi, Gujarati, Malayalam, Punjabi, Assamese, Kannada, Bengali, Tamil, Rajasthani, Haryanvi) using the Moj Multilingual Abusive Comment Identification dataset. While working with multilingual data, apart from a low resource issue, there is a tendency for imbalanced sample distribution. Given that there is a relatively lower number of samples categorized as hateful in less-used languages, it encourages us to adopt transfer learning, data augmentation, and other techniques in AniMOJity.

We base our study on detecting hateful comments for Indian regional languages and analyzing whether they have a biased perspective in the comments towards a particular community. Our main contributions are

- We propose a novel hate speech detection model in a low-resource Indic multilingual setting that incorporates transfer learning and documents the effect of different algorithms on our dataset.

- Our pipeline uses state-of-the-art post-processing techniques to handle hateful behavior by automatically flagging offensive posts.

[*]Equal contribution

- By observing dominant topics (gender, clothing, age, religion, race) in the comments, we manually annotate 300 samples for these bias categories. This dataset broadens the scope of analysis research for social media platforms.

- We further provide a comprehensive analysis using LDA on our dataset to determine the specific keywords and perspectives of these commenters towards the content creators, who are at the brunt of this hate culture.

The remainder of the paper is structured as follows. Section 2 looks at some of the similar works in this domain. Section 3 describes the training dataset, followed by an in-depth presentation of our methodology incorporating data pre-processing and model architecture in Section 4. Next, in Sections 5 and 6, the experimental setup and results are explored, followed by Section 7, where we analyze bias in comments and provide a detailed overview of our findings. We conclude with section 8, discussing future avenues for our proposed model.

2 Related Work

This section briefly sheds light on the various methodologies adopted to tackle hate speech detection and multilingual text classification over the past few years, serving as a benchmark for our research.

Hate Speech Detection Hate speech detection has been at the center of the academic community's attention due to its constantly evolving and picking up different forms with time. The problem categorizes itself as binary or multi-class classification. (Waseem and Hovy, 2016) created a three-class Twitter dataset annotated as sexist, racist, and neutral for offensive language detection. (Kumar et al., 2018) showcased their findings on an aggression identification task discriminating 15,000 annotated Facebook posts and comments in English and Hindi as non-aggressive, covertly aggressive, and overly aggressive. (Davidson et al., 2017) presented a 24,000 corpus for identifying English tweets belonging to profanity, hate speech, and non-offensive categories. (Mandl et al., 2019) gave a detailed account of offensive language identification where three datasets available for Hindi, German, and English were created from Twitter and Facebook. (Zampieri et al., 2019) and (Zampieri et al., 2020) presented their results in several languages obtained from the SemEval competition.

Multilingual Text Classification Multilingual text classification (MTC) aims to breach the language barrier by improving monolingual models by scaling to different languages. (Prajapati et al., 2009) introduced the implementation of translating documents to a universal language for classification, which was bolstered by (Li et al., 2018) to extract grammatical and semantic features from the translated dataset before classification. However, the noise accumulated by translation errors creates a disparity in the final results. (Amini et al., 2010) combined two semi-supervised learning techniques, co-regularization, and consensus-based self-training, to investigate multilingual text classification on a dataset containing five different languages: English, German, French, Italian, and Spanish. (Mittal and Dhyani, 2015) studied MTC in Spanish, Italian, and English by using the N-gram technique and Naïve Bayes to predict the language of a document in classification. (Bentaallah and Malki, 2012) compared two wordnet-based approaches for multilingual text categorization. One relies on the WordNet associated with each language while excluding the translation, and the other focuses on a dis-ambiguation strategy to focus on the most common meaning of the word and access WordNet using a machine translation. Data augmentation ((Ibrahim et al., 2018)) and transfer learning (Roy et al., 2021) help combat situations where there is a lack of training data, both of which are adopted to improve our training data. Recently, promising techniques involving deep learning and contextual embeddings have spearheaded a dynamic shift in the approach to tackling MTC tasks. Transformers became a mainstay in cross-lingual tasks and ushered in mBERT (Devlin et al., 2018), a multilingual masked language model, and XLM (Conneau and Lample, 2019). (Khanuja et al., 2021) proposed a multilingual LM, MuRIL, specifically built for Indic languages.

Impact of Biased Attack on Social Media Biases Make People Vulnerable to Misinformation Spread by Social Media, and cognitive biases originate in how the brain processes the information every person encounters daily. The study by (Döring and Mohseni, 2020) analyses the comments on YouTube and displays a gender bias in the comments. Most attacks are against female content creators and are not just hateful but offensive. Furthermore, the analysis by (Aguirre and Domahidi, 2021) portrays that the comments on YouTube are sexual as well as racist in nature. Thus, biased and offensive comments against them can highly ruin their image. Our work combines pre and post-processing techniques with a novel transfer learning pipeline for hate speech detection in low-resource languages, as well as analyzes the bias in comments against content creators on the Moj platform.

3 Dataset

The dataset utilized in this study was made available by the Moj Multilingual Abusive Comment Identification Challenge organizers in partnership with IIIT-D as part of that challenge [1]. Given the natural language and contextual user data, the project aims to combat abusive comments on Moj, one of India's largest short-form video apps, in thirteen languages, as shown in Figure 1. Figure 2 shows the distribution of which language contains the most hateful comments.

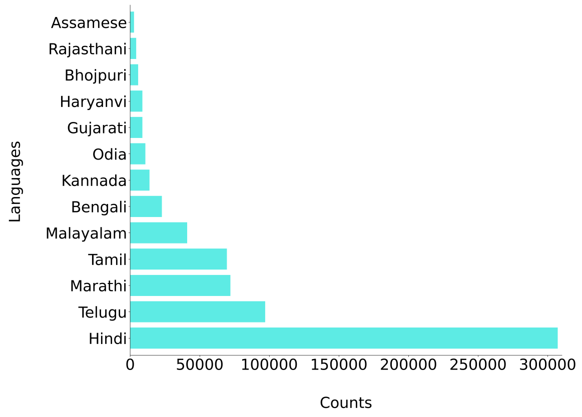


Figure 1: Distribution of languages present in the Moj Multilingual Abusive Comment Identification dataset

3.1 Data Pre-processing

Data preprocessing aims to maintain the input text's original grammatical structure and linguistic information while reducing stop-words, inhibiting the loss of information. To retain key information, we followed the following preprocessing steps

- We created a list of common stop-words to remove from the dataset for each language and converted the text to lowercase. Here stop-words are nugatory words that do not influence the output.
- We substituted emojis by their linguistic meaning in the tweet for each source language. To capture the contextual meaning of an emoji, we tried to incorporate emoji2vec. However, due to the diverse nature of our dataset, apart from a few languages like Hindi, there was not any pre-existing support for languages like Assamese, Gujrati, etc.

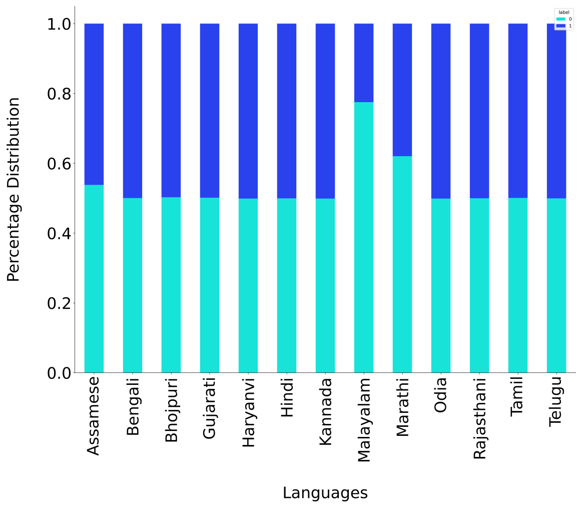


Figure 2: Percentage distribution of Hate vs Non-hate in each language in the Moj Multilingual Abusive Comment Identification dataset

The conventional method of using ekphrasis for preprocessing does not work well with a multilingual dataset, encouraging us to adopt Indic NLP (Kakwani et al., 2020) and NLTK library support for preprocessing Hindi. However, due to the tokenization constraints in Indic NLP, we perform tokenization using XLM-R.

4 Methodology

This section documents the techniques used to achieve the study's main objective.

4.1 XLM-R Model and finetuning

We use XLM-RoBERTa (XLM-R), a universal cross-lingual model trained on 100 different languages, using input ids to determine the language used. One of the critical differences XLM offers over its counterparts is the fact that it uses a stream of an arbitrary number of sentences, truncating the ones exceeding a limit. Unlike some XLM multilingual models, XLM-R does not require language tensors to identify the language used and can determine the correct language from the input id. We

[1]https://www.kaggle.com/competitions/iiitd-abuse-detection-challenge

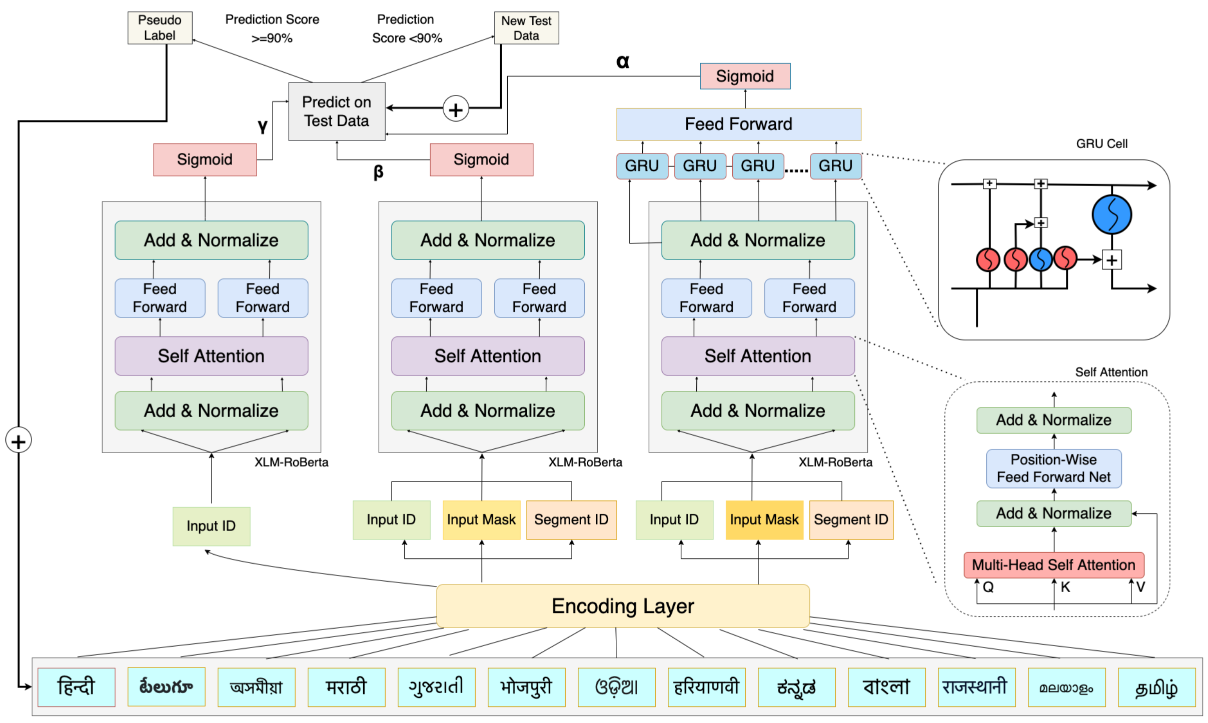


Figure 3: **Illustration of AniMOJity:** It consists of three language models merged together using α, β, γ as weighted parameters

fine-tuned our model by adding layers to the core model using pre-trained artificial neural networks.

4.2 Incorporating MLM with Fine Tuning

In Masked Language Modelling, a fixed percentage of words are masked, and the model is expected to predict the masked words based on the other words. During fine-tuning, the parameters of the pre-trained models are frozen while the detection layer is updated using an optimization algorithm to minimize the loss function.

$$\max_{\theta} \log p_{\theta}(\overline{\mathrm{x}} \mid \hat{\mathrm{x}}) \approx \sum_{t=1}^{T} m_t \log p_{\theta}\left(x_t \mid \hat{\mathrm{x}}\right) \qquad (1)$$

In equation 1, we maximize the probability of a masked token x_t to appear in the t 'th position in a sequence given the tokens in that sequence, x_hat.

4.3 Model Design

The architecture for AniMOJity (Figure 3) is described in this section. We used a combination of three distinct models as described below:

- In our first model, we pass Input-Id as the vectorized input through a pre-trained XLM-R model (Conneau and Lample, 2019). The output (last-hidden-state) obtained from the model is passed as an input to a Dense layer having sigmoid as an activation function and binary cross-entropy as the loss function. Adam optimizer with a learning rate of 5e-6 was used to train the model.

- For our second model, we pass Input-Id, token-type-id, and attention mask as the vectorized input through a pre-trained XLM-R model. The output (last-hidden-state) obtained from the model is passed as an input to the Dense layer having sigmoid as the activation function and binary cross-entropy as the loss function. Adam optimizer with a learning rate of 1e-5 is used to train the model.

- Finally, in the third model, we pass Input-Id, token-type-id, and attention mask as the vectorized input through a pre-trained XLM-R model. The output (last-hidden-state) obtained from the model is passed as an input to the GRU cell (Chung et al., 2014) (having 128 units), and output from the GRU cell is flattened and connected to a Dense Layer having sigmoid as the activation function and binary cross-entropy as the loss function. Adam optimizer with a learning rate of 1e-5 is used to train the model.

Algorithm 1: AniMOJity's Training Algorithm

```
Data: Hateful comment dataset
Result: AniMOJity: A meta-model used to predict hateful comments
Create 3 XLM-R-based Models;
while $I \neq 2$ do
    while $J \neq 3$ do
        Train Model-(j) on Training Dataset and record the result for Test dataset;
        $J \leftarrow J + 1$;
    end
    Predictions on test set (Y-hat) = $\alpha$ * (Predictions from Model-(1)) + $\beta$ * (Predictions from Model-(2)) + $\gamma$ * (Predictions from Model-(3));
    Y-hat (having a confidence score above 90% for hateful and below 10% for not hateful comments) are used as features to create an augmented dataset for the (ith)-level model;
    $I \leftarrow I + 1$;
end
the (2)-level model makes predictions on the test set
```

For model stacking, we combine their predictions to create a model using the fusion weights (α, β, γ) shown in Figure 3. We train multiple base models to predict a target variable while concurrently using the predictions of each model to predict the value of the target variable.

4.4 Psuedo labelling

Pseudo-labeling involves using labeled data to predict unlabelled data. The trained model generates pseudo labels for an unlabelled dataset, combined with the original labels for a final model training. This improves the model's robustness by creating a more precise decision boundary. We implement pseudo-labeling while working on our dataset by utilizing the labels where the predictions on the test set have a confidence score of more than 90% and less than 10% for hateful and inoffensive comments, respectively.

4.5 Text Classification

During text classification, a transformer model takes the final hidden state (h) of the first token [CLS] as the representation of the whole sequence. To classify a comment as hateful or not, we pass the fine-tuned representation of the comment to a sigmoid function (equation 2) and train the model to optimize the binary cross entropy loss (equation 3).

$$\mathrm{p}(\mathrm{c} \mid \mathbf{h}) = softmax(\mathrm{Wh}) \tag{2}$$

Here, W denotes the weights for the classification layer and h is the final hidden state.

$$L_{\mathrm{BCE}} = -\frac{1}{m}\sum_{i=1}^{m}\left(y_{(i)}\log\left(\hat{y}_{(i)}\right) + \left(1 - y_{(i)}\right)\log\left(1 - \hat{y}_{(i)}\right)\right) \tag{3}$$

Where, $y_{(i)}$ and y_t represent the ground truth and predicted class of the ith sample in a dataset. Since binary classification means a class takes either 0 or 1 as its input, if $y_{(i)} = 0$ term ceases to exist, and if $y_{(i)} = 1$ then the $\left(1 - y_{(i)}\right)$ term becomes 0.

5 Experimental Setup

This section elaborates on the baselines, modules, and functions used to construct AniMOJity. As discussed in related work, we used m-BERT (Devlin et al., 2018) as our baseline model, which has been the golden standard for multilingual text classification tasks. Using m-BERT as our backbone, we experimented with different architectures, among which the instances where we found promising results are seen in 1. However, the primary limitation with m-BERT was the lack of support for low-resource languages, which led us towards MuRIL (Khanuja et al., 2021) since it has been trained on a wide assortment of Indian regional languages to improve downstream NLP tasks. However, the most significant update XLM-Roberta offers over a model confined to a limited amount of training data, like MuRIL, is the significantly increased amount of training data, which in conjunction with the Masked Language Modelling approach discussed in Section 4.2, cements it as state of the art.

We first split our task into two pipelines: learning and testing, where the training (learning) process is carried out twice before making predictions on the test set as explained in Algorithm 1 below. We employ two models showing state-of-the-art results on multilingual classification, mBERT, and XLM-R (Conneau and Lample, 2019) as baselines for

Table 1: Performance evaluation of the binary hate speech classification based on Moj Multilingual Dataset for 13 low resource languages in terms of Accuracy and F1 Score

S.No.	Model	Accuracy (%)	Test F1 Score (%)
1	CNN - Single Input (m-Bert)	93.269	87.181
2	CNN - Multiple Input (m-Bert)	93.420	87.449
3	CNN (m-Bert)	92.898	87.232
4	Concat Pooling	94.726	87.933
5	MuRIL	93.30	87.420
6	CNN - Single Input (XLM-R)	93.823	87.619
7	CNN - Multiple Input (XLM-R)	94.333	88.369
8	GRU Cell (XLM-R)	94.628	88.377
9	CNN - Attention (XLM-R)	94.529	88.497
10	CNN - LSTM (XLM-R)	94.240	88.290
11	CNN – BiLSTM (XLM-R)	**96.324**	87.181
12	**AniMOJity**	95.604	**88.602**

Table 2: Hate and Bias-level breakdown of the multi-label 300 annotated samples. Note that we can observe more than one category of bias for a comment

Type	Gender	Clothing	Age	Racial	Religion
Hate	72	148	30	82	65
No Hate	102	26	140	92	109

our binary classification task. While working on these transformer-based models, we came across three types of inputs: input-id, token type id, and attention mask, of which we tested different combinations. The output (last-hidden-state) obtained from these models is passed as an input to the classification head. The models implemented in this study are created using Python 3.10.0 with Tensorflow v2.6.1 as the deep learning framework to build the architecture and train on Graphical Processing Unit (GPU Tesla P100 16GB) servers of Kaggle. To evaluate the performance of our system, we conduct experiments comparing different models. We created and trained our models using TensorFlow and Keras after dividing the dataset into a 90/10 ratio. Accuracy and F1 scores are used as the evaluation criteria. We assess the outcomes of the architecture shown in Table 1.

6 Results

In this section, we describe the evaluation results obtained after testing each language and briefly examine the performance of different models.

Practical Findings By comparing the proposed model with benchmarks, we demonstrate the effectiveness of our architecture. Table 1 shows the investigative analysis of different strategies we used on the multilingual task. After exhaustive experimentation with different architectures, we exhibit the capability of AniMOJity to deal with offensive language detection.

Theoretical Findings Our suggested methodology performed very well when applied to comments where a nasty word or phrase guided the user's intent after using AniMOJity to categorize remarks as offensive or inoffensive. There were, however, a few instances where someone employing a slang phrase or colloquialism in a humorous or referential manner was mistakenly tagged as hateful because of a significant constraint while working on hate speech detection: the accurate classification of "hate."

Analytical Findings We performed an exploratory data analysis of our dataset described in Section 7 using Latent Dirichlet Allocation (LDA) (Jelodar et al., 2019) to understand the latent topics and derive semantic relationships of different themes and trends prevalent in our dataset. For example, a common variety of comments in our dataset shaming the inappropriate clothing style adopted by an influencer on the app led to many off-hand remarks that the model could not correctly identify, leading to a heavy reliance on LDA. This dependency of being familiar with specific topics in the dataset serves as a prospect that a hate speech tagger can incorporate will improve cases where a lack of context may lead to misclassification.

7 Analysing Bias in Comments

Hate comments against content creators that are defined as defamatory statements intended to portray

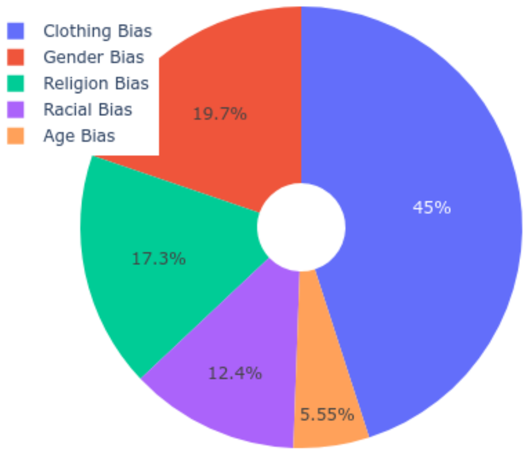


Figure 4: Bias-level percentage distribution of the 300 annotated samples

Table 3: Clothing-level topics obtained from Latent Discriminatory Analysis with their top words

Topic	Top Words
Harassing the influencer based on clothing choice	Kapde (clothes), Pehn (wear), Tarika (style), N*ngi (naked)
Suggesting to change clothing style	video, full, kapde (clothes), pehno (wear)
Implying the influencer wants more followers	Followers, body, like, chahiye (want)
Implying that the content isn't Family friendly	Family, kapde (clothes), utaar (remove), problem
This topic infers the support of others against the negative comment section	comment, gande (bad), apko (you), karte (do)

the artists are unfavorably within the broad category of offensive content on the Internet. A statement is discriminatory if it targets a person belonging to a particular social group segment for discriminatory reasons. For instance, targeting a specific gender, color, or religion might cause bias. To shed light on these biased attacks, we annotate 300 samples described in Table 2, and the distribution of these annotations is shown in Figure 4. For our sample, we randomly select comments labeled as Hate and which are in the Hindi language. While not exhaustive, the manually annotated labels offer a glimpse into the distribution, quality, and quantity of hateful comments. To further our analysis on a granular level, we perform Linear Discriminant Analysis (LDA) on the subsections of each data to identify the targeted sub-topics under the bias categories.

7.1 Gender Bias

Suppose gender bias is predominant in these social media platforms. In that case, it will perpetuate existing stereotypes, necessitating social media platforms to re-examine their algorithms as, ultimately, negatively shaping people's notion of a significant issue. Inferred from our annotation, we found a ratio of 11:1 tweets geared towards female content creators, which leads us to inspect further the subject on which the discrimination is based. For our analysis, we pick three topics to further our study on gender bias; solely gender, clothing, and age.

Clothing Bias Body covering or attire is an integral part of creating a persona that is available for perception by others. Clothing is also one of the most significant indications of gender identity, and being subject to a toxic environment, as portrayed by hate comments, can take a severe mental toll. It is vital to engage in a far more considerable effort to eradicate toxic attitudes learned consciously or unconsciously from mass media's modern 'schools.' Figure 4 shows that defamation against women's clothing is dominant compared to slander solely on gender or a woman's age. While labeling for clothing bias, we searched for clothing-specific keywords, ranging from clothing style to the variations in techniques used. For example, we observe clothing bias in comment; *"Kabhi kapde pahan Kar Bhi video banaa liya karo ladki ke naam per kalank Ho Tum" (Make a video after wearing clothes, you are a stigma in the name of the girl)*, by identifying the word "kapda"(clothes). We observed five major topics prevalent in our dataset along with their top keywords, as shown in Table 3, obtained from topic modeling using LDA.

Solely based on gender To differentiate comments based on their target gender demographic, we analyzed the "gender" attributed to important words. For example, in the comment; *"Saal** dikhawa karti hai suwar" (sister-in-law pretends to be a pig)*, certain words like "karta" (male) and "Karti" (female) both mean "do" / "be" but are gender specific along with certain keywords which are used to refer to women in a derogatory manner (for example *"saal**"* below means sister-in-law via a direct translation, however, it used in a negative connotation).

Age Bias While labeling comments based on age, we searched for specific keywords that negatively refer to someone's age. For example, in analysis; *"Are aunty apne beti ke kpde phn liya kya" (Hey aunty have you put on your daughter's clothes?)*, "aunty" is used to refer to elderly women.

Still, it can be used negatively depending on the words used alongside it. Despite the limited age bias in our sample size, we can't ignore the fact that some comments target a content creator's age. We are further cementing the gender bias discussed above. The top words obtained from the sample dataset after performing topic modeling were: Aunty(aunt), Maa(mother), Chudail(witch), and Bhuda(old), which in itself shows three out of four words directly targeting elder female content creators.

7.2 Racial Bias

In order to determine racial bias in the comments, we searched for keywords referring to the color of an individual's skin (for example, kala/kali- is used to address someone with a darker skin tone, and similarly, gora/gori is used to address someone with a fair skin tone). Topic modeling was used to obtain the following top words: kali(black), gori(white), chipkali(lizard), kalank(tainted). Based on the keyword search, most of the hateful comments associated with a racial bias had a close correlation with an individual's attire (clothing bias) or the public reception towards their body. In an example comment; *"Pari nahi tu kali chudail h apne mann me hi pari banti firti" (You are not a fairy, you're a black witch. You're only a fairy in your dreams)*, the word "kali (black)" is used in a negative light; attacking someone on racial grounds.

7.3 Religion Bias

Based on the results obtained from topic modeling, we could see a strong correlation between religious bias, clothing bias, and female bias, as many comments undermined women based on their religious affiliation and their choice of clothing. The top words observed were: Mulla(Muslim), Hindu(Hinduism), Sardar(Sikh), and Islam(Muslim). An Example:*"Aap musalman hokar bhi aise kapde pahnati ho kuchh to sharm karo adla pakshi" (Even though you are a Muslim, you wear such clothes, you should be ashamed.)*

8 Conclusion & Future Work

Any negative statement based on identification (such as gender, caste, or religion) rather than comments supporting the formation of an inclusive community should be avoided. To achieve this goal, we test various deep learning approaches on the Moj Multilingual Abusive Comment Identification dataset having thirteen distinct regional languages and constructed a model that outperforms our baselines. To advance our research, we manually annotate 300 data points with bias labels (gender, clothing, age, religion, race). A common thread that ties together other hateful comments and biases observed is the reference to clothes and how people perceive clothes as being inappropriate. Suppose we follow this through-line of hate geared towards the choice of clothing. In that case, the fact that most hateful comments are targeted towards female influencers hearkens to a societal issue of objectification and dehumanization that makes women prone to attacks and libel.

The model architecture can be improved in the future by testing other feature selection methods, elevating its overall performance while working with code-mixed languages. Second, research has shown the importance of context for hate speech classification. Certain cases arise where there is a lack of contextual information, causing our best model to misclassify specific entries where even humans would struggle. This may be mitigated by developing a more robust pipeline by incorporating steps such as co-reference analysis and sarcasm detection. Third, to comprehend where the model fails, there is a need for a detailed investigation of false positives and negatives. Furthermore, the research may be further carried out to analyze bias on other social media platforms.

9 Ethical Statement

Using datasets and algorithms for hate speech detection can have beneficial and harmful effects. We want to be clear that our intention is not to advance any discourse (biased or otherwise). Instead, by providing a more balanced real-world view of the discussion against content creators in India, we hope to educate the audience about the distorted commentary perspectives in India. Through research and analysis in this area, we hope to create more reliable platforms for discussing creators on social media.

References

Luis Aguirre and Emese Domahidi. 2021. Problematic content in spanish language comments in youtube videos about venezuelan refugees and migrants. *Journal of Quantitative Description: Digital Media*, 1.

Massih R Amini, Cyril Goutte, and Nicolas Usunier. 2010. Combining coregularization and consensus-based self-training for multilingual text categorization. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 475–482.

Mohamed Amine Bentaallah and Mimoun Malki. 2012. The use of wordnets for multilingual text categorization: A comparative study. In *ICWIT*, pages 121–128.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. *Advances in neural information processing systems*, 32.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the international AAAI conference on web and social media*, volume 11, pages 512–515.

Ona De Gibert, Naiara Perez, Aitor García-Pablos, and Montse Cuadros. 2018. Hate speech dataset from a white supremacy forum. *arXiv preprint arXiv:1809.04444*.

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*.

Nicola Döring and M. Rohangis Mohseni. 2020. Gendered hate speech in youtube and younow comments: Results of two content analyses. *Studies in Communication and Media*.

Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)*, 51(4):1–30.

Mai Ibrahim, Marwan Torki, and Nagwa El-Makky. 2018. Imbalanced toxic comments classification using data augmentation and deep learning. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)*, pages 875–878. IEEE.

Hamed Jelodar, Yongli Wang, Chi Yuan, Xia Feng, Xiahui Jiang, Yanchao Li, and Liang Zhao. 2019. Latent dirichlet allocation (lda) and topic modeling: models, applications, a survey. *Multimedia Tools and Applications*, 78(11):15169–15211.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, NC Gokul, 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.

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*.

Ritesh Kumar, Atul Kr Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking aggression identification in social media. In *Proceedings of the first workshop on trolling, aggression and cyberbullying (TRAC-2018)*, pages 1–11.

Ximing Li, Changchun Li, Jinjin Chi, Jihong Ouyang, and Chenliang Li. 2018. Dataless text classification: A topic modeling approach with document manifold. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 973–982.

Thomas Mandl, Sandip Modha, Prasenjit Majumder, Daksh Patel, Mohana Dave, Chintak Mandlia, and Aditya Patel. 2019. Overview of the hasoc track at fire 2019: Hate speech and offensive content identification in indo-european languages. In *Proceedings of the 11th forum for information retrieval evaluation*, pages 14–17.

S Mittal and P Dhyani. 2015. Multilingual text classification. *Int. J. Eng. Res. Technol.(IJERT)*, 4(3).

Bhagirath P Prajapati, Sanjay Garg, and Mahesh H Panchal. 2009. Automated text categorization with machine learning and its application in multilingual text categorization. In *National Conference on Advance Computing-NCAC09, Vallabh Vidyanagar, Anand, Gujarat, India*, pages 204–209.

Björn Ross, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurowsky, and Michael Wojatzki. 2017. Measuring the reliability of hate speech annotations: The case of the european refugee crisis. *arXiv preprint arXiv:1701.08118*.

Sayar Ghosh Roy, Ujwal Narayan, Tathagata Raha, Zubair Abid, and Vasudeva Varma. 2021. Leveraging multilingual transformers for hate speech detection. *arXiv preprint arXiv:2101.03207*.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). *arXiv preprint arXiv:1903.08983*.

Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. Semeval-2020 task 12: Multilingual offensive language identification in social media (offenseval 2020). *arXiv preprint arXiv:2006.07235*.

A Appendix

A.1 Dataset

Moj[2] is India's largest short-video app for multiple regional languages. We chose to use Moj as the basis of our study primarily because it facilitates the use of numerous regional languages and, in doing so, captures the sentiments of different communities at a granular level which cant be achieved with other social media platforms. Another factor that we weighed heavily is that because Moj was recently introduced, its hate flagging capabilities are not as well developed as the internationally established social media platforms, where content moderation removes extremely hateful comments and doesn't accurately depict how a community can spread online hate.

The Moj Multilingual Abusive Comment dataset provided by IIIT-D had the following characteristics:

- The human-annotated dataset was split into two sections: training and testing, each with 665k and 74k samples, respectively.
- The distribution of Abusive and Not Abusive samples was 312k and 352k, respectively.
- All the comments in the dataset are annotated according to the language used. There are instances where similar words in Hindi are code-mixed to create two variants based on the script used, namely Devanagari and Roman-Hindi. Similarly, regional languages like Marathi, Haryanvi, and Rajasthani are variants of the Devanagiri script, were code mixed with English, along with the other regional languages that follow their script (Kannada, Malayalam and Odia-Brahmi, Bengali-Bangla, Bhojpuri-Kathi, Tamil, Telugu-Abugida script a variant of Brahmi Script).
- The test dataset used in this research was not disclosed to the competitors and is not publically available as it was a part of the Moj Multilingual Abusive Comment Identification Challenge hosted by IIIT-D [3].

A.2 Training Strategy

- We noticed a modest difference between GPU and TPU accelerators: models trained on GPU perform significantly better. However, because the experimental time on TPU was shorter, we decided to use it for most of our trials which can be seen in Table 4.

Table 4: Time taken for each Epoch in hours

Info	XLM-Roberta
Accelerator	Tesla P100
Time Taken (hr)	11.54
EPOCH 1	3.86
EPOCH 2	3.72
EPOCH 3	3.49

- We also tried truncation sizes of 64, 128, and 256 and settled on 128 for the input text.
- We used different alpha, beta, and gamma values based on the test f1 score of each model to assign a higher weightage to the model that performed better. On conducting an exhaustive analysis of different combinations of alpha, beta, and gamma, we concluded that our model performed the best for the values of 0.35,0.33,0.34.
- Post-processing: Based on our findings, raising the threshold offered us an advantage. Thus we chose to adjust the thresholds for each language. After experimenting with various thresholds, we discovered that the numbers in Table 5 produced the best results.

[2]https://apps.apple.com/in/app/moj-short-video-app/id1523457550
[3]https://www.kaggle.com/competitions/iiitd-abuse-detection-challenge

Table 5: Language Wise Inference Threshold

Language	**threshold**
Marathi	0.56
Malayalam	0.52
Hindi	0.58
Telugu	0.62
Tamil	0.51
Odia	0.4
Gujarati	0.5
Bhojpuri	0.52
Haryanvi	0.6
Assamese	0.55
Kannada	0.5
Rajasthani	0.5
Bengali	0.55

- Psuedo Labelling: We continued the process of training our model over the course of two iterations, reintegrating examples from the test dataset that provided a prediction probability of greater than 90% to our training dataset giving us a boost of 1% in the test F1 score.

A.3 Annotation Guidelines

Due to the Hindi language's highest density and annotator proficiency, we employed stratified sampling to create a sample size of 300 randomly selected comments in the language. We examined the effects of various biases on the classification of hate using this sample as our starting point. Due to the dataset's limited annotation for hate categorization in the competition, we had three undergraduate students annotate each comment in accordance with the following guidelines: like a comment would be considered biased based on religion if it contained words relating to identifying a person based on their religion like; "Islam", "Islaam","Molla", "Mulla", "Muslim", "Musalman", "Isai", "Christ", "Singh", "Sardar", etc. Similarly, the details for the other categories are provided while analysis in Section 7.

Revisiting Anwesha: Enhancing Personalised and Natural Search in Bangla

Arup Das
IIT Madras, India,
cs20s016@smail.iitm.ac.in

Joyojyoti Acharya
IIT Madras, India,
cs21m024@smail.iitm.ac.in

Bibekananda Kundu
CDAC Kolkata, India,
bibekananda.kundu@gmail.com

Sutanu Chakraborti
IIT Madras, India,
sutanuc@cse.iitm.ac.in

Abstract

Bangla is a low-resource, highly agglutinative language. Thus it is challenging to facilitate an effective search over Bangla documents. We have created a gold standard dataset containing query document relevance pairs for evaluation purposes. We utilise Named Entities to improve the retrieval effectiveness of traditional Bangla search algorithms. We suggest a reasonable starting model for leveraging implicit preference feedback based on the user search behaviour to enhance the results retrieved by the Explicit Semantic Analysis (ESA) approach. We use contextual sentence embeddings obtained via Language-agnostic BERT Sentence Embedding (LaBSE) to rerank the candidate documents retrieved by the traditional search algorithms (tf-idf) based on the top sentences that are most relevant to the query. This paper presents our empirical findings across these directions and critically analyses the results.

1 Introduction

Owing to India's multilingual diversity, it is important to ensure that a wide gamut of people from diverse backgrounds are able to access the web without any language barrier. Bengali alternatively known as Bangla, has 300 million speakers globally and has witnessed the fastest growth among the other Indic languages in terms of internet usage (KPMG, 2017). Hence there is a pressing need to develop tools that facilitate semantic search over Bangla text documents. Previously, efforts have been put into creating Bangla Search engines. For example, Anwesan[1] was built to search over Rabindra Rachanabali collection[2] (Das et al., 2012). Sandhan[3] is a monolingual domain-specific search engine limited to tourism and health domains in nine Indian languages: Bangla, Hindi, Marathi, Tamil, Telugu, Punjabi, Odiya, Gujarati and Assamese. It is based on the Bag of Words model and focuses more on improving recall than precision (Priyatam et al., 2012). Hence the top results are not always relevant for the query. Pipilika[4], launched on April 13, 2013, is designed for the residents of Bangladesh. It crawls data from Bangla News, Bangla Blogs and Bangla Wikipedia. However, Anwesan and Pipilika[5] are not presently accessible for exploration. This paper reports follow-up work based on (Das et al., 2022) recent work, which introduced an exploration toward building অন্বেষা (Anwesha), a prototype for a search engine in Bangla. Anwesha demonstrated promise in addressing the existing search engines' shortcomings and advanced the research done in the information retrieval (IR) space for the Bangla language. Anwesha incorporated the use of diverse knowledge sources like IndoWordNet[6] (Bhattacharyya, 2010), statistical co-occurrences (by way of Latent Semantic Analysis (LSA) (Deerwester et al., 1990)) and external knowledge sources like Wikipedia (by way of Explicit Semantic Analysis (ESA) (Gabrilovich and Markovitch, 2007)) for facilitating effective retrieval, opening gateways to further improvements in the search quality results. The authors have released a Gold Standard dataset[7] containing 94 query doc-

[1] http://anwesan.iitkgp.ernet.in/
[2] https://rabindra-rachanabali.nltr.org/
[3] http://sandhan.tdil-dc.gov.in/Search
[4] https://pipilika.com/
[5] https://en.wikipedia.org/wiki/Pipilika
[6] The official website and the web interface of IndoWordNet: https://www.cfilt.iitb.ac.in/indowordnet/
[7] https://doi.org/10.5281/zenodo.6583149

ument relevance pairs over a test collection of 1182 documents. The collection contains 182 short stories, novels and essays written by Rabindranath Tagore[8] and 1000 newspaper articles published in 2013 crawled from the daily newspaper of Bangladesh Prothom Alo[9]. Every query was designed to belong to one of the four different complexity levels, as shown in Table 1. By designing queries of different complexity levels, the effectiveness of Anwesha as the queries become more difficult to resolve could be studied. An approach like tf-idf works best on precise queries that directly match the content of the relevant documents (complexity level 1). For queries which were not precise, query expansion techniques using IndoWordnet helped make a lexical search like tf-idf perform effectively (complexity level 2). LSA performs well when the query and the retrieved relevant documents do not share many words in common; instead, they share a common theme (complexity levels 3 and 4). ESA performs well when the queries require external background knowledge for their intent resolution (complexity level 4). The queries created with different complexity levels demonstrated that there is no silver bullet which works the best across all types of queries. Each of the top ten documents retrieved by their search algorithm was assigned a score of 1 if irrelevant, 2 if partially relevant, and 3 if completely relevant by at least five Bangla users. Further, Anwesha explains the search results by highlighting words from the documents LSA or ESA reckon to be semantically related to the query.

In its present form, Anwesha has been evaluated on a small set of queries. The lack of handling of multi-word expressions has adversely affected Anwesha's performance in several cases, especially in complexity level 1 query. It does not use the implicit feedback the user provides via click preferences to improve the retrieval effectiveness. For best results, users are restricted by the choice of words that exactly match the contents of relevant documents. Any change in word order in the query can potentially lead to contrasting results, which are not captured in Anwesha.

[8]https://rabindra-rachanabali.nltr.org/

[9]https://www.prothomalo.com/

This paper presents four directions to address the current limitations of Anwesha. First, we expand the Gold Standard dataset by creating an additional 100 query document relevance pairs over a new test collection[10] of 1000 documents for a more exhaustive evaluation and better analysis. Second, we identify the technical terms and named entities in the documents and queries apart from considering only uni-gram word tokens as was done previously. Third, we introduce a novel approach to improve the effectiveness of the IR system by incorporating implicit preference feedback via clickthrough data in an ESA setting. Lastly, we present the usage of contextual vector representations of documents and query using Language-agnostic BERT Sentence Embedding (LaBSE) (Feng et al., 2020) to rerank the documents based on the best sentences from the document that are relevant to the query. We believe that our approaches can be adapted for other low-resource, highly inflected and agglutinative languages similar to Bangla, such as Assamese, Maithili, Oriya and Manipuri (Ray et al., 1966).

The rest of this paper is organized as follows. In Section 2, we position our approaches in the context of background work and relevant research. Section 3 describes the implementation details of our approaches. Section 4 presents a critical analysis of our empirical findings and observations. Section 5 summarizes our key contributions and discusses potential extensions of the work.

2 Background and Literature Survey

This section discusses the concepts that will be used in the rest of the paper.

2.1 Multiword Expressions (MWEs)

MWEs are frequently repeating idiosyncratic phrasal units exhibiting varying degree of semantic compositionality (Chakraborty et al., 2014) (Dandapat et al., 2006). Identifying MWEs is known to play an important role in understanding natural language queries which in turn helps in improving the retrieval effectiveness (Acosta et al., 2011). In the present work, we focus on extracting only the Named Entities (NEs) like names of people (রবীন্দ্রনাথ ঠাকুর (EN: Rabindranath

[10]https://zenodo.org/record/7376906

Query Type	Complexity Level
The query contains exact words, phrases or sentence from the document.	1
The query is not present as it is in the document. There is a slight deviation.	2
The query is a generalised phrase capturing the overall story or the document's theme.	3
It is a general query not related to any specific document.	4

Table 1: Definition of the complexity level of a query

Thakur)), names of locations (পোর্ট ব্লেয়ার (EN: Port Blair)), names of organisations (নাসিরাবাদ পলিটেকনিক ইনস্টিটিউট (EN: Nasirabad Polytechnic Institute)) etc.

In order to detect a multiword NE token in a document or query we used IndicNER (Arnav Mhaske, 2022). IndicNER[11] was trained on the largest publicly available NE Annotated dataset for Indic languages (961679 training instances in the case of Bangla) while the one devised by Sagor Sarker (Sarker, 2021) was trained on a smaller dataset (64155 sentences).

2.2 Explicit Semantic Analysis (ESA)

ESA exploits knowledge of Wikipedia. Terms and documents are expressed in terms of underlying interpretable concepts, where each concept corresponds to a Wikipedia article name. There is an overlap between the concepts shared by similar query terms. For example, the query terms "ক্যান্সার" (EN: cancer) and "কার্সিনোমা" (EN: carcinoma) share লিউকোমিয়া (EN: Leukemia), বায়োপসি (EN: biopsy), সার্ভিকাল (EN: cervical) and ম্যালিগন্যান্ট (EN: malignant) as top concepts. This helps in retrieving relevant documents even when they do not contain the query terms.

2.3 Implicit Preference Feedback

When the information needed is tacit, and the user cannot express her intent, we often do not expect the user to reformulate the query from scratch on search failure. Further, diverse intents can give rise to the same query. Such challenges make it difficult to arrive at an appropriate query representation in the concept space of ESA. However, it becomes easy to implicitly refine the query by unobtrusively studying the user's preference of documents to a query by assessing their interaction with the IR system. Therefore, it is reasonable to engage in implicit iterative query refinement by analyzing the documents selected by the user. The implicit preference feedback system assumes that clicking on a document and viewing it indicates the user's interest in the document's contents (White et al., 2002). Viewing some documents may lead users to refine their understanding of the information they seek and help disambiguate their search requirements (Manning et al., 2008). Explicit feedback can be substituted with implicit feedback in web-based IR, and such feedback represents the user preferences reasonably accurately (Joachims et al., 2005).

2.4 Usage of Language Models for low-resource IR

Queries written in natural language enable better search results when the system can take into account the order of the words in the sentences. Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) is a state-of-the-art approach to produce contextual embeddings that have outperformed previously existing methods like Word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), ELMo (Peters et al., 2018) and ULMFiT (Howard and Ruder, 2018) in tasks like question answering, sentence pairs similarity, sentence pair completion, named entity recognition, entailment classification, sentiment classification and several others because of its unsupervised and deeply bidirectional approach. Search engines like Google (Nayak, 2019) and Bing (Zhu, 2019) have been using BERT to understand the context of the query intent better and allow users to ask questions in a way humans ask experts. However, for low-resource languages like Bangla, where limited data is available, traditional vector space approaches like tf-idf or BM25[12] are preferred as these algorithms are computationally less intensive and do not require additional training

[11]https://huggingface.co/ai4bharat/IndicNER

[12]https://kmwllc.com/index.php/2020/03/20/understanding-tf-idf-and-bm-25/

data (Lin, 2019). Therefore, to benefit from the best of both worlds, we utilise BERT, as illustrated in Section 2.5, as a re-ranker over the top documents retrieved by the tf-idf vector space approach.

2.5 Ad Hoc document retrieval with BIRCH

As per a study (Qiao et al., 2019), MS MARCO[13] passage ranking is closer to the seq2seq task because of its question-answering focus and so pre-trained contextual models like BERT can perform well on them. But for TREC-style ad-hoc document retrieval tasks, we need to fine-tune on user clicks, and the surrounding context is not enough. Further, BERT was not trained with an objective to perform inference on long documents. A simple solution presented by the authors of BIRCH (Akkalyoncu Yilmaz et al., 2019) is to perform sentence-level inference on a candidate document and pick the best sentences (in practice three most relevant sentences) or paragraph in a document which will act as an appropriate proxy for document relevance. This approach, in one way, is a form of passage retrieval, where BERT has already been studied to perform well. The final score of a document to a query is as follows:

$$S_f = a \cdot S_{doc} + (1 - a) \cdot \Sigma_{i=1}^{n} w_i \cdot S_i \quad (1)$$

S_f is the final document score obtained using the BIRCH approach, S_{doc} is the original document score as per a traditional retrieval algorithm like BM25 or tf-idf, S_i is the i^{th} best sentence identified by BERT, a and w_i's are hyperparameters, tuned as per the parameters that gave the highest average precision (AP) score on the training folds.

2.6 Sentence Embeddings using LaBSE

A commonly used approach to obtain a sentence embedding from a BERT Base model is to average the BERT output layer (768 dimensions) or use the output of the [CLS] token from the last transformer block. However, these standard approaches often produce sentence embeddings, that are even worse than those obtained by averaging GloVe embeddings (Reimers and Gurevych, 2019). To obtain good sentence embeddings, we need to fine-tune the BERT output. IndicBERT produces multilingual word embeddings for 12 Indian languages (including Bangla) (Kakwani et al., 2020). LaBSE is a BERT multilingual embedding model developed by Google that generates cross-lingual sentence embeddings for 109 languages (including Bangla). It is trained on 17 billion monolingual sentences and 6 billion bilingual sentence translation pairs using Masked Language Modelling (MLM) and Translation Language Modelling (TLM) pre-training. The empirical benefits in terms of its effectiveness in diverse tasks including retrieval are analysed in (Feng et al., 2020). The sentence embeddings are obtained using l_2 normalized [CLS] token representations from the last transformer block.

3 Proposed Methodology

3.1 Identifying NEs in MWEs

We quantitatively analyzed the NEs detected by (Arnav Mhaske, 2022) and (Sarker, 2021) on two NE recognition datasets ((Karim et al., 2019) and (Pan et al., 2017)). We observed that the NEs detected by the latter were a subset of the NEs detected by IndicNER. Hence, to obtain the multiword NE tokens in a document or query, we used IndicNER.

3.2 Implicit Preference Feedback Strategy

The algorithm for implicit preference feedback with ESA is as follows:

- *Step 1*: The user issues a query.
- *Step 2*: The system returns the top retrieved documents based on the cosine similarity between the vector representation of the document and the query in the concept space.
- *Step 3*: The user clicks and views some of the retrieved documents.
- *Step 4*: The system promotes and demotes the top concepts of the query present in the documents visited by the user and documents viewed by the user but not visited, respectively. The weights for the top concepts of the query, which were common to both the highly preferred and less preferred documents, remain unaffected.
- *Step 5*: The system displays a revised set of retrieved results.
- *Step 6*: Repeat steps 3,4,5 until the user views no new interesting documents.

[13]https://microsoft.github.io/msmarco/

Implicit feedback inference can be made only on the documents the user has observed and assessed. A simple strategy studied in (Radlinski and Joachims, 2005) is adopted to determine the documents observed by a user. As per the study, a user generally follows the results from top to bottom and mostly observes the results from the document at rank one to the document below the one clicked and viewed by the user. A user at least looks at the top two results with equal attention. However, the user is more likely to click on the first result. So if a user only visits the first document presented in the retrieved result, it was assumed that the user had assessed only the first and second documents in the top retrieved documents. Hence the weight updates in the query's concept vector representation were made for the top two retrieved documents. For two documents, both visited, the document visited later should be given a higher preference (Joachims et al., 2005).

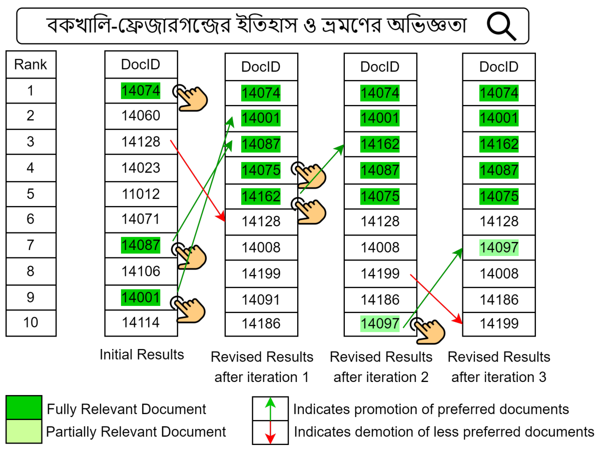

বকখালি-ফ্রেজারগঞ্জের ইতিহাস ও ভ্রমণের অভিজ্ঞতা
Rank
DocID
14074
14060
14128
14023
11012
14071
14087
14106
14001
14114
14074
14001
14087
14075
14162
14128
14008
14199
14091
14186
14074
14001
14162
14087
14075
14128
14008
14199
14186
14097
14074
14001
14162
14087
14075
14128
14097
14008
14186
14199
Initial Results
Revised Results after iteration 1
Revised Results after iteration 2
Revised Results after iteration 3
Fully Relevant Document
Partially Relevant Document
Indicates promotion of preferred documents
Indicates demotion of less preferred documents

Figure 1: Top 10 retrieved results with implicit preference feedback after every iteration.

In Figure 1, we present the top retrieved results after every iteration of query refinement using the implicit preference feedback from the user for the query "বকখালি-ফ্রেজারগন্জের ইতিহাস ও ভ্রমণের অভিজ্ঞতা"/bakakhāli-phrejāraganjera itihāsa ō bhramaṇera abhijñatā (EN: History and Travel Experience of Bakkhali-Fraserganj). We can observe that with every iteration, the vector representation of the query gets closer to the relevant documents in the concept space of ESA; as a result, the number of relevant documents in the top ranks increases. Some of the less preferred documents occupy lower ranks in the top results, while many of them disappear from the retrieved list.

3.3 Application of BERT (LaBSE) for document retrieval using BIRCH

IndicBert was trained on MLM task with a word-level objective using a cross-entropy loss function. In contrast, LaBSE was trained on MLM and TLM tasks with a sentence-level objective using additive margin softmax as a contrastive loss function. Since a dual encoder model is trained using a translation ranking loss, the similarity or dissimilarity of sentences in a shared embedding space is more adequately captured by LaBSE than IndicBERT, which uses a single BERT model. Thus, LaBSE can better discriminate amongst the most similar sentences. This was also confirmed through our empirical findings. Therefore we have used LaBSE to obtain the sentence embeddings in the BIRCH approach. We find an initial pool of top 150 candidate documents using the tf-idf approach. We used LaBSE to find the best three sentences that resolve the query in every candidate document. We used a convex combination (as in Equation (1)) of the LaBSE inference scores with retrieved scores from tf-idf to obtain the final document scores. In the original work, the optimal values for a and w_i's in Equation (1) were obtained by fine-tuning using an exhaustive grid search approach. Since we do not have enough data to fine-tune the hyperparameters, we give equal importance to the keyword-based approach (query-document cosine similarity based on tf-idf approach) and to the aggregated sentence level evidence obtained through LaBSE. In the absence of a large amount of training data, we use ESA scores as surrogates to guide the selection of w_i values since ESA scores are expected to correspond to a human assessment of similarities based on familiar background concepts. Interestingly, as highlighted in Section 4.4, this has been empirically found to consistently improve retrieval effectiveness compared to a scheme where the top three sentences are weighed equally. Hence $w_i =$ cosine similarity score of the query and sentence S_i for a candidate document d_j in the concept space defined by ESA. All embeddings for sentences in documents are available as part of pre-computation. Only query embeddings are created at runtime. Such a standard approach boosts time efficiency of retrieval[14].

4 Results and Analyses

4.1 Gold Standard Dataset Preparation

The Gold Standard dataset created by (Das et al., 2022) contained documents from the prominent dialect variations in Bangla: Sadhu Bhasa[15] and Chalit Bhasa[16]. In view of contributing to the linguistic diversity of the existing test collection, we curated a fresh dataset of 1000 text documents which consists of 642 documents for West Bengal Bangla news readers of Ebela, Zee News and Anandabazar Patrika (Kunchukuttan et al., 2020). One hundred twenty-eight news articles belong to the Entertainment, International and National category, while 129 news articles belong to the Kolkata and sports category. One hundred forty-five articles on the health-specific domain were obtained from Vikaspedia[17], an online information guide launched by the Government of India. The remaining 213 articles were based on the travel domain and were crawled from various Bangla travel blogs. We designed 100 queries each belonging to one of the four complexity levels in Table 1. We obtained graded relevance feedback from at least five Bangla annotators on the top ten documents retrieved by our search algorithms for a given query. A document was rated 1 if irrelevant, 2 if partially or reasonably relevant, and 3 if completely relevant to the query. The final relevance of a document to a query is the mean of the user's relevance scores. Table 2 presents the statistics of the documents in our test collection.

4.2 MWE Evaluation with NEs

We take twenty queries each belonging to one of the four complexity levels (Table 1) from the dataset in (Das et al., 2022) and our Gold Standard dataset to evaluate the effectiveness of grouping NE tokens with respect to mean normalized discounted cumulative gain (nDCG) and mean average precision (MAP). We present our results in Table 3. Using NE-enabled search has boosted the retrieval performance on both datasets i.e., (Das et al., 2022) and our dataset.

Earlier, for the queries like “বাংলাদেশ ইনস্টিটিউট অব ব্যাংক ম্যানেজমেন্ট”(EN: Bangladesh Institute of Bank Management), the documents with a high presence of the individual tokens “বাংলাদেশ” (EN: Bangladesh), “ইনস্টিটিউট” (EN: Institute), “অব” (EN: of), “ব্যাংক” (EN: Bank) and “ম্যানেজমেন্ট” (EN: Management) were prioritised over documents having the query words as a single unit. A query may not precisely contain the named entities as it is present in the relevant documents. So we indexed both the uni-gram tokens and the multiword NE tokens. The revised tokens formed for the example query are: “বাংলাদেশ ইনস্টিটিউট অব ব্যাংক ম্যানেজমেন্ট” (EN: Bangladesh Institute of Bank Management), “বাংলাদেশ” (EN: Bangladesh), “ইনস্টিটিউট” (EN: Institute), “অব” (EN: of), “ব্যাংক” (EN: Bank) and “ম্যানেজমেন্ট” (EN: Management).

4.3 Implicit Preference Feedback Results

To fully utilise the benefit of ESA, it is necessary that the concepts chosen are representative of the underlying text semantics. Since Bangla is a resource-constrained language, we could not find enough relevant concepts about health and travel from Wikipedia alone. We supplemented this gap with articles from reliable sources like Vikaspedia and various travel blogs. We represented the complete test collection using 9349 articles. Due to the absence of articles related to the literary works of Rabindranath Tagore, we have not used ESA on the 182 documents from Rabindranath Tagore's work.

We present the performance of Implicit preference feedback on 40 queries in Figure 2. We observed that implicit feedback on the initial results for queries from complexity levels 1 and 2 did not generate any interesting document in the revised result set. We expected this behaviour as the queries were clear and precise in intent and did not require any refinement in its vector representation. On the average, queries from complexity levels 3 and 4 produced improved results after

[14]https://github.com/huggingface/transformers/issues/876#issuecomment-514948425
[15]https://en.banglapedia.org/index.php/Sadhu_Bhasa
[16]https://en.banglapedia.org/index.php/Chalita_Bhasa
[17]https://bn.vikaspedia.in/

Parameters	Entire Test Collection	Entertainment	Health	International	Kolkata	National	Sports	Travel
Tokens (words)	433553	34629	91084	29506	25182	30699	31005	191444
Types (unique words)	53831	9707	14650	9436	7671	8927	8599	22622
Sentences	34553	3644	6439	2879	2814	2730	3214	12833
Average number of sentences per document	34.553	28.468	44.715	22.492	21.813	21.328	24.914	59.967
Average number of tokens per sentence	12.547	9.503	14.145	10.248	8.948	11.245	9.648	14.918
Average number of tokens per document	433.553	270.539	632.527	230.515	195.209	239.835	240.379	894.598

Table 2: Statistics of our Gold Standard Dataset

Dataset	Model	Mean nDCG@10	MAP@10	Mean Precision@10
(Das et al., 2022)	tf-idf	0.741	0.453	0.62
	tf-idf + NE tokens	0.842	0.519	0.66
Ours	tf-idf	0.76	0.57	0.37
	tf-idf + NE tokens	0.92	0.8	0.49

Table 3: Performance of Anwesha across different query complexity levels containing NEs.

1 and 2.3 iterations of query refinement using implicit preference feedback. Complexity level 4 queries required at most three iterations of query refinement. These queries were not a precise articulation of query intent. Hence, the query was initially not well represented in the concept space. After the user implicitly provided the IR system with the click-through information, the concept representation of the query improved.

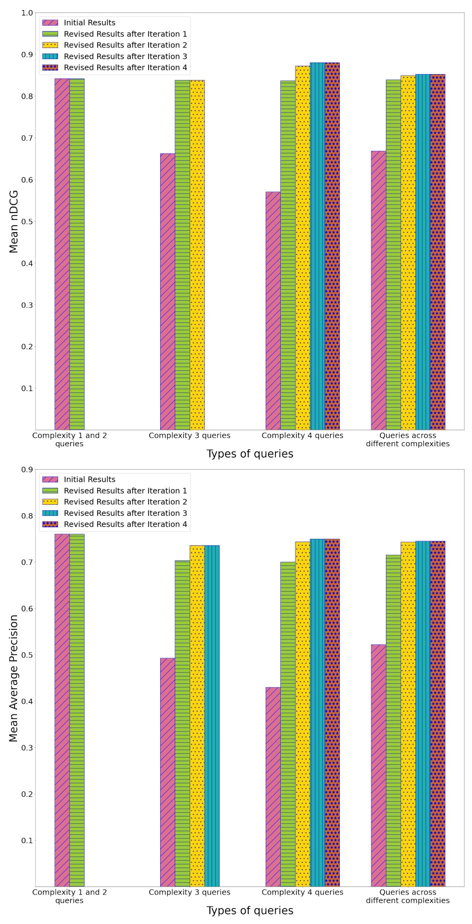


Figure 2: Results of Implicit Preference Feedback across different query complexity levels measured using mean nDCG and MAP @K = 10.

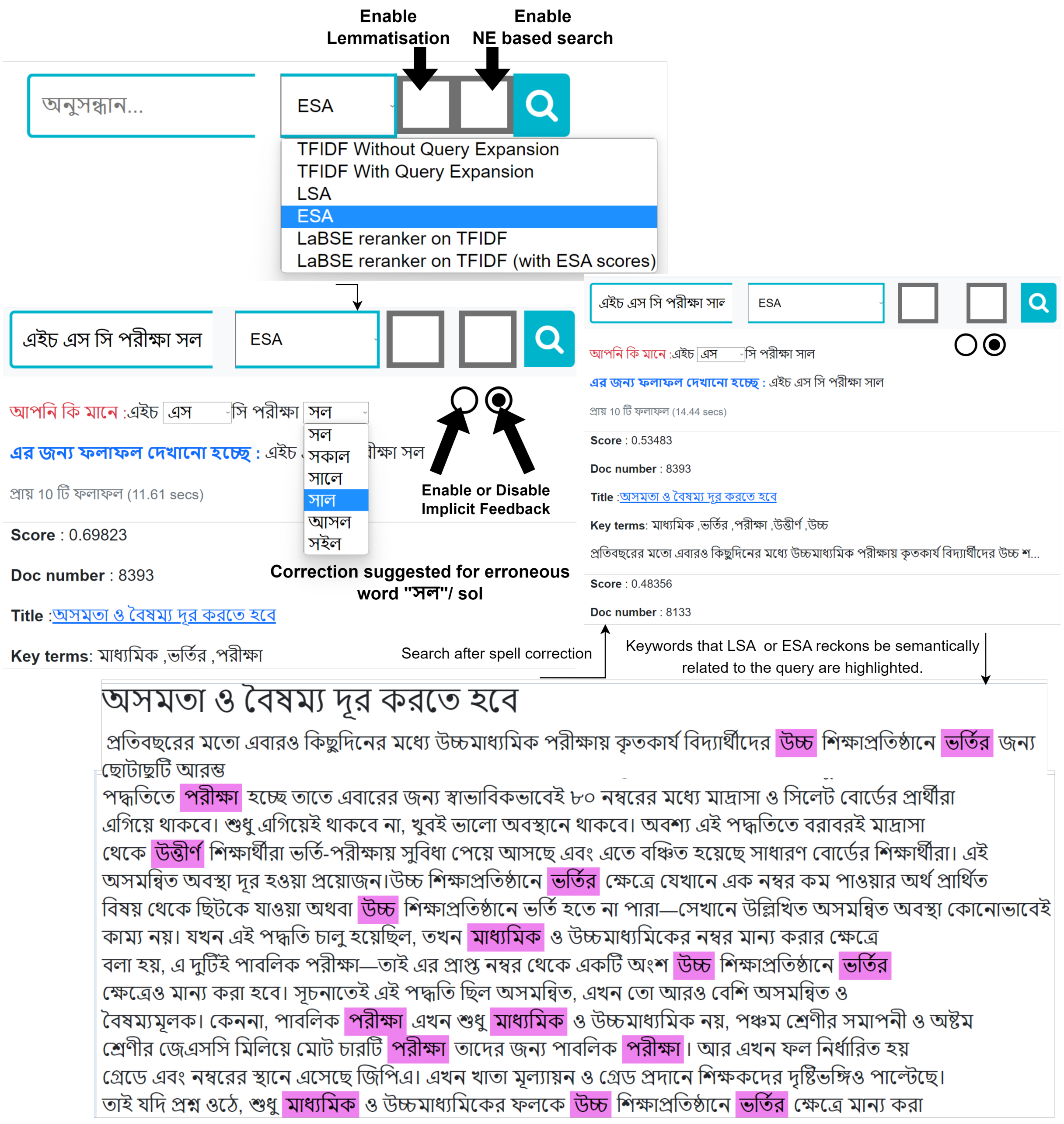
4.4 Evaluation of LaBSE reranker using BIRCH approach

We present the results of applying LaBSE to document retrieval using the BIRCH approach in Table 4. We have studied the effect of LaBSE reranker on candidate documents retrieved by the tf-idf vector space algorithm with uniform weights ($w_1 = w_2 = w_3 = 1$) and weights set to cosine similarity score of the best sentences with the query in the concept space of ESA. Interestingly, we observe ESA weighted sentence scores perform the best. We present the top two documents retrieved using BIRCH in Figure 4 and the most relevant sentences related to the query "যমুনা নদীর দক্ষিণে সপ্তম আশ্চর্য্য"/ yamunā nadīra dakṣiṇe saptama āścaryya (EN: Seventh wonder to the south of Yamuna river). The query contains geospatial details that LaBSE could capture. So it picked the sentences from the documents related to the Taj Mahal with such information. The same query was issued in Sandhan (as of: 1-Oct-2022) with the intent to seek documents related to the "Taj Mahal"; the top 10 documents retrieved by Sandhan are not in line with the goal. Due to the Bag of words nature of Sandhan, the documents that contain a high presence of the individual terms "দক্ষিণ"/ dakṣiṇa(EN: south), "নদী"/ nadī(EN: river) and "যমুনা"/ yamunā(EN: Yamuna) are deemed to be given higher preference. The semantic relationships among the query words are not captured.

Figure 3 shows a snapshot of the revised user interface of Anwesha where the user can choose one of the six options (tf-idf, IndoWordNet based query expansion, LSA, ESA, LaBSE reranker on tf-idf with uniform sentence weights and LaBSE reranker on tf-idf with sentence weighted by ESA scores) for re-

Domain	Model	Mean nDCG@10	MAP@10	Mean precision@10
Travel	tf-idf	0.811	0.69	0.535
	ESA	0.824	0.716	0.56
	LaBSE reranker on tf-idf	0.908	0.82	0.635
	LaBSE reranker on tf-idf weighted by ESA scores	0.952	0.889	0.69
Health	tf-idf	0.772	0.621	0.38
	ESA	0.826	0.704	0.4
	LaBSE reranker on tf-idf	0.878	0.777	0.46
	LaBSE reranker on tf-idf weighted by ESA scores	0.912	0.825	0.47

Table 4: Performance of LaBSE as a reranker on tf-idf retrieved candidate documents.


Enable Lemmatisation
Enable NE based search
অনুসন্ধান...
ESA
TFIDF Without Query Expansion
TFIDF With Query Expansion
LSA
ESA
LaBSE reranker on TFIDF
LaBSE reranker on TFIDF (with ESA scores)
এইচ এস সি পরীক্ষা সল
সল
সকাল
সালে
সাল
আসল
সইল
Enable or Disable Implicit Feedback
Correction suggested for erroneous word "সল"/ sol
Search after spell correction
Keywords that LSA or ESA reckons be semantically related to the query are highlighted.

আপনি কি মানে :এইচ এস সি পরীক্ষা সল

এর জন্য ফলাফল দেখানো হচ্ছে : এইচ এস সি পরীক্ষা সল

প্রায় 10 টি ফলাফল (11.61 secs)

Score : 0.69823

Doc number : 8393

Title :অসমতা ও বৈষম্য দূর করতে হবে

Key terms: মাধ্যমিক ,ভর্তির ,পরীক্ষা

এইচ এস সি পরীক্ষা সাল

আপনি কি মানে :এইচ এস সি পরীক্ষা সাল

এর জন্য ফলাফল দেখানো হচ্ছে : এইচ এস সি পরীক্ষা সাল

প্রায় 10 টি ফলাফল (14.44 secs)

Score : 0.53483

Doc number : 8393

Title :অসমতা ও বৈষম্য দূর করতে হবে

Key terms: মাধ্যমিক ,ভর্তির ,পরীক্ষা ,উত্তীর্ণ ,উচ্চ

প্রতিবছরের মতো এবারও কিছুদিনের মধ্যে উচ্চমাধ্যমিক পরীক্ষায় কৃতকার্য বিদ্যার্থীদের উচ্চ শ...

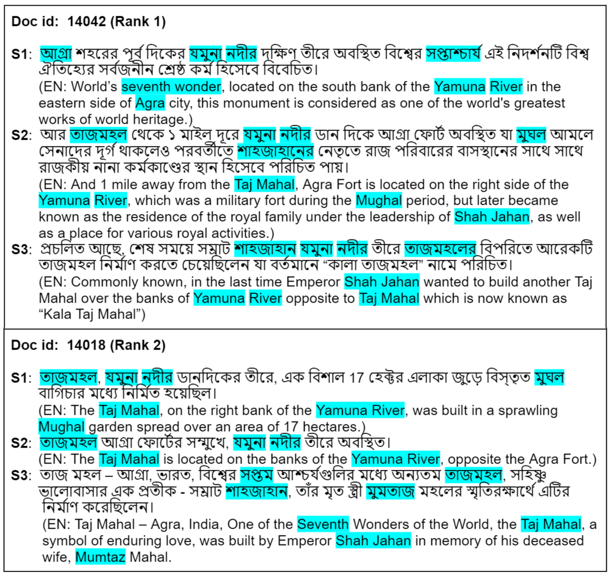
Score : 0.48356

Doc number : 8133

অসমতা ও বৈষম্য দূর করতে হবে

প্রতিবছরের মতো এবারও কিছুদিনের মধ্যে উচ্চমাধ্যমিক পরীক্ষায় কৃতকার্য বিদ্যার্থীদের উচ্চ শিক্ষাপ্রতিষ্ঠানে ভর্তির জন্য ছোটাছুটি আরম্ভ
পদ্ধতিতে পরীক্ষা হচ্ছে তাতে এবারের জন্য স্বাভাবিকভাবেই ৮০ নম্বরের মধ্যে মাদ্রাসা ও সিলেট বোর্ডের প্রার্থীরা এগিয়ে থাকবে। শুধু এগিয়েই থাকবে না, খুবই ভালো অবস্থানে থাকবে। অবশ্য এই পদ্ধতিতে বরাবরই মাদ্রাসা থেকে উত্তীর্ণ শিক্ষার্থীরা ভর্তি-পরীক্ষায় সুবিধা পেয়ে আসছে এবং এতে বঞ্চিত হয়েছে সাধারণ বোর্ডের শিক্ষার্থীরা। এই অসমন্বিত অবস্থা দূর হওয়া প্রয়োজন।উচ্চ শিক্ষাপ্রতিষ্ঠানে ভর্তির ক্ষেত্রে যেখানে এক নম্বর কম পাওয়ার অর্থ প্রার্থিত বিষয় থেকে ছিটকে যাওয়া অথবা উচ্চ শিক্ষাপ্রতিষ্ঠানে ভর্তি হতে না পারা—সেখানে উল্লিখিত অসমন্বিত অবস্থা কোনোভাবেই কাম্য নয়। যখন এই পদ্ধতি চালু হয়েছিল, তখন মাধ্যমিক ও উচ্চমাধ্যমিকের নম্বর মান্য করার ক্ষেত্রে বলা হয়, এ দুটিই পাবলিক পরীক্ষা—তাই এর প্রাপ্ত নম্বর থেকে একটি অংশ উচ্চ শিক্ষাপ্রতিষ্ঠানে ভর্তির ক্ষেত্রেও মান্য করা হবে। সূচনাতেই এই পদ্ধতি ছিল অসমন্বিত, এখন তো আরও বেশি অসমন্বিত ও বৈষম্যমূলক। কেননা, পাবলিক পরীক্ষা এখন শুধু মাধ্যমিক ও উচ্চমাধ্যমিক নয়, পঞ্চম শ্রেণীর সমাপনী ও অষ্টম শ্রেণীর জেএসসি মিলিয়ে মোট চারটি পরীক্ষা তাদের জন্য পাবলিক পরীক্ষা। আর এখন ফল নির্ধারিত হয় গ্রেডে এবং নম্বরের স্থানে এসেছে জিপিএ। এখন খাতা মূল্যায়ন ও গ্রেড প্রদানে শিক্ষকদের দৃষ্টিভঙ্গিও পাল্টেছে। তাই যদি প্রশ্ন ওঠে, শুধু মাধ্যমিক ও উচ্চমাধ্যমিকের ফলকে উচ্চ শিক্ষাপ্রতিষ্ঠানে ভর্তির ক্ষেত্রে মান্য করা

Figure 3: Revised user interface of Anwesha and explanation of search results by highlighting keywords


Doc id: 14042 (Rank 1)

S1: আগ্রা শহরের পূর্ব দিকের যমুনা নদীর দক্ষিণ তীরে অবস্থিত বিশ্বের সপ্তাশ্চার্য এই নিদর্শনটি বিশ্ব ঐতিহ্যের সর্বজনীন শ্রেষ্ঠ কর্ম হিসেবে বিবেচিত।
(EN: World's seventh wonder, located on the south bank of the Yamuna River in the eastern side of Agra city, this monument is considered as one of the world's greatest works of world heritage.)

S2: আর তাজমহল থেকে ১ মাইল দূরে যমুনা নদীর ডান দিকে আগ্রা ফোর্ট অবস্থিত যা মুঘল আমলে সেনাদের দূর্গ থাকলেও পরবর্তীতে শাহজাহানের নেতৃতে রাজ পরিবারের বাসস্থানের সাথে সাথে রাজকীয় নানা কর্মকাণ্ডের স্থান হিসেবে পরিচিত পায়।
(EN: And 1 mile away from the Taj Mahal, Agra Fort is located on the right side of the Yamuna River, which was a military fort during the Mughal period, but later became known as the residence of the royal family under the leadership of Shah Jahan, as well as a place for various royal activities.)

S3: প্রচলিত আছে, শেষ সময়ে সম্রাট শাহজাহান যমুনা নদীর তীরে তাজমহলের বিপরিতে আরেকটি তাজমহল নির্মাণ করতে চেয়েছিলেন যা বর্তমানে "কালা তাজমহল" নামে পরিচিত।
(EN: Commonly known, in the last time Emperor Shah Jahan wanted to build another Taj Mahal over the banks of Yamuna River opposite to Taj Mahal which is now known as "Kala Taj Mahal")

Doc id: 14018 (Rank 2)

S1: তাজমহল, যমুনা নদীর ডানদিকের তীরে, এক বিশাল 17 হেক্টর এলাকা জুড়ে বিস্তৃত মুঘল বাগিচার মধ্যে নির্মিত হয়েছিল।
(EN: The Taj Mahal, on the right bank of the Yamuna River, was built in a sprawling Mughal garden spread over an area of 17 hectares.)

S2: তাজমহল আগ্রা ফোর্টের সম্মুখে, যমুনা নদীর তীরে অবস্থিত।
(EN: The Taj Mahal is located on the banks of the Yamuna River, opposite the Agra Fort.)

S3: তাজ মহল – আগ্রা, ভারত, বিশ্বের সপ্তম আশ্চর্যগুলির মধ্যে অন্যতম তাজমহল, সহিষ্ণু ভালোবাসার এক প্রতীক - সম্রাট শাহজাহান, তাঁর মৃত স্ত্রী মুমতাজ মহলের স্মৃতিরক্ষার্থে এটির নির্মাণ করেছিলেন।
(EN: Taj Mahal – Agra, India, One of the Seventh Wonders of the World, the Taj Mahal, a symbol of enduring love, was built by Emperor Shah Jahan in memory of his deceased wife, Mumtaz Mahal.

Figure 4: Best three sentences (S1, S2, S3) considered relevant by LaBSE in the top two retrieved documents ranked by the BIRCH approach. The words in a sentence deemed relevant to the query intent resolution by ESA are highlighted by the system.

trieval. The user can disable sending implicit feedback to the IR system, apply lemmatisation and NE based search. Anwesha receives the query "এইচ এস সি পরীক্ষা সল"/ ēica ēsa si parīkṣā sala(EN: HSC Exam Year) and performs spelling correction on the query word সল/ sala → সাল/ sāla. The search results are explained by highlighting the words relevant to the query in a top retrieved document. Such explanations are useful for techniques like LSA and ESA where the documents not having the query words are retrieved.

5 Conclusion and Future Work

We have enabled NE-based search to obtain single-unit NE tokens. To the best of our knowledge, ours is the first effort that delivers personalized results from diverse background knowledge sources via the clickthrough information of the user. We are also not aware of the past work that applies BERT models and it's advancement on Bangla IR. We have extended the previously existing Gold standard dataset for diverse evaluation of search results and used this to systematically study the improvement of Anwesha while addressing its current limitations. Our technique can inspire research in IR for other low-resource, highly inflected languages. As part of future work, we plan to handle different forms of MWEs like conjunct verbs (example "অনুভব করা"/ anubhaba karā (EN: to feel)), noun-verb collocations (example "খেতে যাওয়া"/khete yāōẏā (EN: go eat)), reduplicated terms (example "ছোট ছোট"/choṭa choṭa (EN: small small)), idiomatic compound nouns (example: "ভাই বোন"/bhāi bona (EN: brother sister)) etc (Chakraborty et al., 2014) (Dandapat et al., 2006). In future, we can incorporate actions like bookmarking, saving a document, and time of viewing a document as other contributors to determining user preferences. ESA is a recall-centric approach while LaBSE, as a reranker on the tf-idf vector space approach, is precision oriented. Studying the effect of the ESA cosine similarity scores used as surrogates to weigh the LaBSE sentence scores will be interesting. In some cases, where the queries were not precise, tf-idf could not include the relevant candidate documents in the initial pool of candidates. So LaBSE could not perform well on them, suggesting that a recall-centric algorithm could plausibly be used to determine the initial candidate set. Estimating the sentence-level relevance of a document to the query is a reasonable approach because LaBSE is trained with a sentence-level objective, making it suitable to find the relationship between a query sentence and a sentence from the document. However, in this approach, we are losing context information. Hence a more robust language model that could encode the "long" documents while effectively gauging their relevance to a "short" query would be a better alternative to our existing approach using BIRCH. In future, we plan to conduct our experiments over a wider range of queries. The dataset and code of our work are present here: `https://github.com/ArupDas15/Revisiting_Anwesha`.

Acknowledgements

This work would not have been possible without the contributions of the members of the Artificial Intelligence and Database Lab affiliated with the Department of Computer Science and Engineering, Indian Institute of Technology Madras Lokasis Ghorai and Arjun Kumar Gupta in the early stages of development of Anwesha.

References

Otavio Acosta, Aline Villavicencio, and Viviane Moreira. 2011. Identification and treatment of multiword expressions applied to information retrieval. In *Proceedings of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World*, pages 101–109, Portland, Oregon, USA. Association for Computational Linguistics.

Zeynep Akkalyoncu Yilmaz, Shengjin Wang, Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Applying BERT to document retrieval with birch. 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): System Demonstrations*, pages 19–24, Hong Kong, China. Association for Computational Linguistics.

Rudramurthy. V Anoop Kunchukuttan Pratyush Kumar Mitesh Khapra Arnav Mhaske, Harshit Kedia. 2022. Naamapadam: A large-scale named entity annotated data for indic languages.

Pushpak Bhattacharyya. 2010. Indowordnet. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, pages 3785–3792.

Tanmoy Chakraborty, Dipankar Das, and Sivaji Bandyopadhyay. 2014. Identifying bengali multiword expressions using semantic clustering. *ArXiv*, abs/1401.6122.

Sandipan Dandapat, Pabitra Mitra, and Sudeshna Sarkar. 2006. Statistical investigation of bengali noun-verb (nv) collocations as multi-word-expressions. *Proceedings of Modeling and Shallow Parsing of Indian Languages (MSPIL)*, pages 230–233.

Arup Das, Bibekananda Kundu, Lokasis Ghorai, Arjun Kumar Gupta, and Sutanu Chakraborti. 2022. Anwesha: A tool for semantic search in bangla. `https://github.com/ArupDas15/Anwesha`. Accepted in The International Conference on Agglutinative Language Technologies as a challenge of Natural Language Processing; Conference date: 07-06-2022 Through 08-06-2022.

Suprabhat Das, Shibabroto Banerjee, and Pabitra Mitra. 2012. Anwesan: A search engine for bengali literary works. *World Digital Libraries*, 5(1):11–18.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2020. Language-agnostic BERT sentence embedding. *CoRR*, abs/2007.01852.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification.

Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, page 154–161, New York, NY, USA. Association for Computing Machinery.

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*.

Redwanul Karim, M. A. Muhiminul Islam, Sazid Rahman Simanto, Saif Ahmed Chowdhury, Kalyan Roy, Adnan Al Neon, Md Hasan, Adnan Firoze, and Rashedur M. Rahman. 2019. A step towards information extraction: Named entity recognition in bangla using deep learning. *J. Intell. Fuzzy Syst.*, 37:7401–7413.

KPMG. 2017. Indian languages- defining india's internet. `https://assets.kpmg/content/dam/kpmg/in/pdf/2017/04/Indian-languages-Defining-Indias-Internet.pdf`, Accessed: 2022-09-09.

Anoop Kunchukuttan, Divyanshu Kakwani, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. 2020. Ai4bharat-indicnlp corpus: Monolingual corpora and word embeddings for indic languages. *arXiv preprint arXiv:2005.00085*.

Jimmy Lin. 2019. The neural hype and comparisons against weak baselines. *SIGIR Forum*, 52(2):40–51.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

P. Pandurang Nayak. 2019. Understanding searches better than ever before. `https://blog.google/products/search/search-language-understanding-bert/`. Accessed: 2022-08-03.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. 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.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations.

Pattisapu Nikhil Priyatam, Srikanth Reddy Vaddepally, and Vasudeva Varma. 2012. Domain specific search in indian languages. In *Proceedings of the First Workshop on Information and Knowledge Management for Developing Region*, IKM4DR '12, page 23–30, New York, NY, USA. Association for Computing Machinery.

Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2019. Understanding the behaviors of bert in ranking.

Filip Radlinski and Thorsten Joachims. 2005. Query chains: Learning to rank from implicit feedback. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM.

Punya Sloka Ray, Muhammad Abdul Hai, and Lila Ray. 1966. *Bengali Language Handbook*. Center for Applied Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Sagor Sarker. 2021. Bnlp: Natural language processing toolkit for bengali language. *ArXiv*, abs/2102.00405.

Ryen White, Ian Ruthven, and Joemon Jose. 2002. The use of implicit evidence for relevance feedback in web retrieval. pages 93–109.

Jeffrey Zhu. 2019. Bing delivers its largest improvement in search experience using azure gpus. `https://azure.microsoft.com/en-us/blog/bing-delivers-its-largest-improvement-in-search-experience-using-azure-gpus/`. Accessed: 2022-08-21.

KnowPAML: A Knowledge Enhanced Framework for Adaptable Personalized Dialogue Generation Using Meta-Learning

Aditya Shukla[§][*]
TCS Research, India
adityashukla5597@gmail.com

Zishan Ahmad[§]
IIT Patna
1821cs18@iitp.ac.in

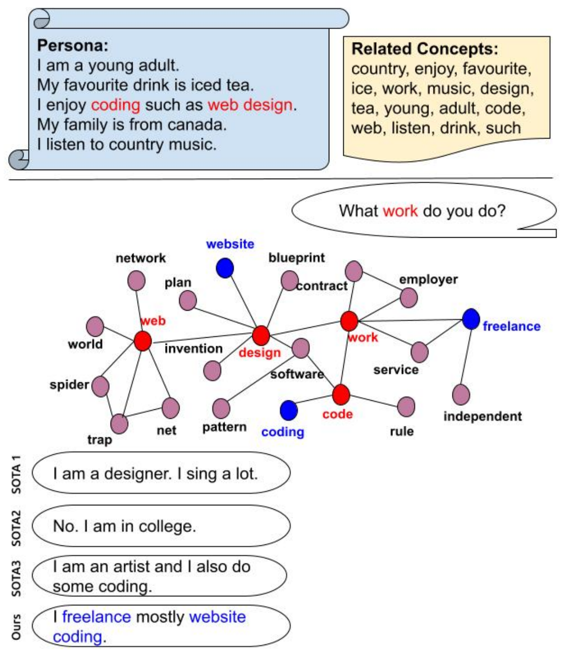
Asif Ekbal
IIT Patna
asif@iitp.ac.in

Abstract

In order to provide personalized interactions in a conversational system, responses must be consistent with the user and agent persona while still being relevant to the context of the conversation. Existing personalized conversational systems increase the consistency of the generated response by leveraging persona descriptions, which sometimes tend to generate irrelevant responses to the context. To solve this problems, we propose to extend the persona-agnostic meta-learning (PAML) framework (Madotto et al., 2019) by adding knowledge from ConceptNet knowledge graph (Speer et al.) with multi-hop attention mechanism (Tran and Niedereée, 2018). Knowledge is a concept in a triple form that helps in conversational flow. The multi-hop attention mechanism helps select the most appropriate triples with respect to the conversational context and persona description, as not all triples are beneficial for generating responses. The Meta-Learning (PAML) framework allows quick adaptation to different personas by utilizing only a few dialogue samples from the same user. Our experiments on the Persona-Chat dataset show that our method outperforms in terms of persona-adaptability, resulting in more persona-consistent responses, as evidenced by the entailment (Entl) score in the automatic evaluation and the consistency (Con) score in human evaluation.

1 Introduction

Recent advancements in personalized dialogue generation techniques that incorporate the personality of the speakers have enabled more human-like, natural, and persona-consistent responses. However, most methods require persona information in the form of style, persona profile, or persona statements, such as "I love meeting new people" and "Autumn is my favorite season" which can be very diverse and hence require a lot of data to model any persona type.



Persona:
I am a young adult.
My favourite drink is iced tea.
I enjoy coding such as web design.
My family is from canada.
I listen to country music.

Related Concepts:
country, enjoy, favourite, ice, work, music, design, tea, young, adult, code, web, listen, drink, such

What work do you do?

I am a designer. I sing a lot.

No. I am in college.

I am an artist and I also do some coding.

I freelance mostly website coding.

Figure 1: An example of developing a persona adaptable and knowledge guided response from test data using meta-learning and a commonsense knowledge graph. Concepts in red nodes come from the persona statement and dialogue history, whereas concepts in blue are in the generated response.

The Persona Agnostic Meta-Learning model (PAML) (Madotto et al., 2019) was developed to deal with these practical problems. This model, trained using meta-learning, would be able to adapt rapidly to new and unseen personas using only a few samples. The popular Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017) framework served as the foundation for the PAML framework. Customized Model Agnostic Meta-Learning (CMAML) (Song et al., 2019) framework and Generating Personalized Dialogue via Multi-Task Meta-Learning (Lee et al., 2021) both largely follow the PAML framework except for an extra network structure optimization component and persona reconstruction component respectively. The Per-

[*]Work done at IIT Patna as part of M.Tech
[§]equal contribution

sonaChat corpus (Zhang et al., 2018), a popular personalized dialogue generating corpus that includes persona statements describing each speaker in addition to persona-specific dialogues, was used to benchmark all these frameworks. However, a comprehensive evaluation of these frameworks reveals that they continue to struggle to respond to situations and do not precisely match the persona statements. For example, in Figure 1, speaker1 asks a question about the nature of work that speaker2 is doing based on the persona statement: *I enjoy coding such as web design*. The responses generated by the State of the art frameworks are insufficiently precise. SOTA1, which is trained using dialogue history and persona statements and tested as a PAML framework, fails to accurately understand the persona. SOTA2, similar to SOTA1, is only trained on dialogue history and hence fails to generate the correct response. SOTA3 is a PAML framework that uses a meta-learning mechanism that wrongly understands the persona. Neither approach appears to be capable of accurately comprehending the persona.

Our model's goal is to appropriately interpret the context and persona statement while generating an engaging response that is both context and persona consistent. Real-life conversations, in general, begin with one topic and transition to other based on the personalities of the speakers. Basic commonplace knowledge also tends to spark conversations. Previously, external knowledge was used as a foundation for research, like "an open-domain knowledge graph" (Xing et al., 2017), "a commonsense knowledge base" (Zhou et al., 2018a), or "background documents" (Zhou et al., 2018b). Such external knowledge is incorporated into this research by enhancing the entity representations in dialogues with it and then generating responses based on it. This enhanced encoding of information in the model improves the quality of the generated responses. Also, commonsense knowledge-grounded (Majumder et al., 2020), knowledge expansion (Zhang et al., 2019), approaches were researched earlier, which also demonstrates an improvement in response generation.

In this work, we enhance the entity representation in persona-profile by using ConceptNet (Speer et al.) triples. We obtain triple representation using a graph encoding technique and incorporate these into the PAML framework (Madotto et al., 2019) using attention mechanism with the attention mechanism (Bahdanau et al., 2014). We then perform experiments on the PERSONA-CHAT dataset (Zhang et al., 2018) to test the effect of knowledge on persona adaptability. The PAML framework focuses on learning the different personas as independent tasks using the meta-learning approach, as opposed to building the model to represent all of the personas. The model is intended to be a few-shot learner using the PERSONA-CHAT where the train-test split contains a non-overlapping persona-type. The meta-learning training steps ensure that the model can swiftly adjust to a new unseen persona profile as well as the reaction style of a certain persona by utilizing only a few dialogues for training. Incorporating commonsense knowledge helps in a deeper understanding of the persona and dialogue context, therefore helping in adaptability.

The response generated by our model (c.f. Figure 1) is pretty much accurate and understands the persona statement correctly. Here, We extracted concepts from persona statement and dialogue context. For example Figure 1, "work" which is extracted from dialogue context is related to "freelance". Similarly, "web" (extracted from persona statement), "design" and "work" all three are related to "freelance" in some way, which is utilised to generate the response. Our experiment results show that our approach is effective in both automatic and human evaluation when compared to baseline models.

2 Related Work

Meta Learning The approach of teaching the model how to learn fast and effectively is known as meta-learning. Before, meta-learning was used in applications such as image classification. Finn et al. (2017) introduced MAML (Model-Agnostic Meta-Learning) technique, and it performed well for few-shot image classification. However, following MAML, various methodologies for NLP applications such as machine translation (Gu et al., 2018) and dialogue generation (Qian and Yu, 2019; Huang et al., 2020; Mi et al., 2019) were proposed, indicating an improvement. And following this, PAML (Madotto et al., 2019) was introduced, with a focus on personalized dialogue generation and Lee et al. (2021) proposed MTML and AMTML two frameworks in which they merged persona reconstruction task with the PAML which improved the persona consistency but failed in other automatic evaluation metrics.

Personalized dialogue generation have caught the interest of many in recent years, following Zhang et al. (2018)'s study of the task with the Persona-Chat dataset. Recent research focuses on advancing the dialogue generation by grounding persona information (Mazaré et al., 2018; Bao et al., 2019; Wolf et al., 2019) or bringing external knowledge such as the knowledge graphs (Long et al., 2017; Ghazvininejad et al., 2018) or supplementary texts (Vougiouklis et al., 2016; Xu et al., 2017) into the model. This research demonstrates that by doing so, the model becomes more informative and consistent with the personas of the speakers, hence improving generation performance.

Furthermore, if the knowledge graph is properly formed, or if it is domain-specific (Zhu et al., 2017; Xu et al., 2017), or if the knowledge base is large enough (Zhou et al., 2018a), the rich semantics representation is included through entities and relations (Hayashi et al., 2020). Based on this, Majumder et al. (2020) grounded the expanded persona statements using a commonsense knowledge graph, which aids in controlling the flow of the conversation. And In addition, reinforcement-learning-based framework was also proposed by Song et al. (2020) and Li et al. (2019) for making dialogue generation more informal. However, all of this work was based on either directly conditioning the response with the persona or by incorporating some commonsense knowledge into the model. None of the works attempted to generate a response from the persona except Madotto et al. (2019) and with the adaption of the knowledge graph.

3 Methodology

Our method extends the PAML framework (Madotto et al., 2019) by utilizing commonsense knowledge to generate personalized dialogues. The PAML is adapted from the MAML, which is capable of quickly adjusting to new, unknown tasks that were not employed during training. We continue to use the PERSONA-CHAT dataset (Zhang et al., 2018), which was used in the PAML framework. The dialogue in PERSONA-CHAT includes the utterances $u_{1:m}$ and persona statements $p_{1:n}$. In previous research, response $R = u_m$ was conditioned on the persona sentences $P = p_{1:n}$ and previous utterances $U = u_{1:m-1}$ following Equation 1:

$$f_W(R|U, P; W) = p(u_m|u_{1:m-1}, p_{1:n}; W) \quad (1)$$

Algorithm 1 KnowPAML

```
Require: $P_m^{train}$, $P_m^{valid}$
Require: Hyperparameters $\eta_{inner}$, $\eta_{outer}$
Require: iteration, patience, count
  Randomly initialize parameter W
  while $count < patience$ do
    persona batch from train set $P_{m_i}^t \sim P_m^{train}$
    for all $P_{m_i}^t$ do
      $(T_{p_i}, V_{p_i}) \sim P_{m_i}^t$
      calculate total loss
      $L_{T_{p_i}}^{total} = L_{T_{p_i}}^g(f_W) + L_{T_{p_i}}^t(f_W)$
      evaluate $\nabla_W L_{T_{p_i}}^{total}(f_W)$
      update $W_{p_i}' = W - \eta_{inner}\nabla_W L_{T_{p_i}}^{total}(f_W)$
    end for
    $W \leftarrow W - \eta_{outer}\nabla_W \frac{1}{S}\sum_{P_{m_i}} L_{V_{p_i}}(f_{W_{p_i}'})$
    persona batch from valid set
    $(T_{p_i'}, V_{p_i'}) \sim P_m^{valid}$
    if iteration % 10 == 0 then
      do for loop as above with $(T_{p_i'})$
      if $L_{V_{p_i}}(W') < L_{V_{p_i}}(W')_{best}$ then
        save weights
      else
        count+=1
      end if
    end if
  end while
```

In PAML, they first adapt W from the set of dialogue created by a persona and then respond conditioned only on the dialogue history rather than conditioning on both the dialogue history and persona sentences. So in this case Eq.(1) becomes:

$$f_W(R|U; W) = p(u_m|u_{1:m-1}; W) \quad (2)$$

3.1 Knowledgeable Persona-Agnostic Meta Learning

We use the PERSONA-CHAT dataset with ConceptNet triples associated with it at the utterance level (Section 3.2). First, we define P_m, which contains all of the personas and divide it into the train P_m^{train}, valid P_m^{valid}, and test P_m^{test} sets in the same way as PAML does. We sample persona batch $P_{m_i}^t$ from P_m^{train} for every training epoch, and then sample a set of utterances and associated triples as training T_{p_i} and another set as validating V_{p_i} from each persona in $P_{m_i}^t$. The utterance history in the batch is passed through a Transformer encoder and a representation $H = [h_1, h_2, ..., h_n]$

is obtained. Two weight matrices $W_{concept\_emb}$ and $W_{relation\_emb}$ are trained along with the model. The head and tail entity indexes are multiplied with $W_{concept\_emb}$ to obtain h_{emb} and t_{emb} respectively. The relation indexes of the triples are multiplied with $W_{relation\_emb}$ to obtain the representation r_{emb}. The final triple representation is obtained by concatenating these three representations $Trip = [h_{emb} : r_{emb} : t_{emb}]$. In this manner with each utterance we finally get a list of triple representations $T = [Trip_1, Trip_2, ..., Trip_n]$. Not all the triples in the list are useful for generating the appropriate knowledge-grounded response. Therefore we need to only select the triples appropriate with respect to the conversational context. To make this selection, we make use of the multi-hop attention mechanism (Tran and Niedereée, 2018). The attention mechanism works on a query q and an input sequence $T = [Trip_1, Trip_2, ..., Trip_n]$. For each k in K hop attention, the following steps are executed:

$$s_t^{(k)} = tanh(W_q^{(k)} Trip_t) \odot tanh(W_g^{(k)} g^{(k-1)}) \quad (3)$$

$$\alpha^{(k)} = softmax(w_s^{(k)^T} s_t^{(k)}) \quad (4)$$

$$o_q^{(k)} = \sum_t \alpha_t^{(k)} Trip_t \quad (5)$$

Here, $W_q^{(k)}$, $W_g^{(k)}$ and $w_s^{(k)}$ are the trainable parameters, and m is a separate memory vector for guiding the next attention step. It is recursively updated using the following equation:

$$g_q^{(k)} = g_q^{(k-1)} + o_q^k \odot q \quad (6)$$

The initial value of vector $g^{(0)}$ is defined based on the context vector $o_q^{(0)}$, given by the equation 7:

$$o_q^{(0)} = \frac{1}{l} \sum_t h_q(t) \odot q \quad (7)$$

The representation $o_q^{(k)}$ is the final attended and summed representation of T. We experiment with two settings for fusing this representation with the encoder hidden representation H: (i: AKnowPAML). At each k the representation $o_q^{(k)}$ is added to each step of the encoded representation H to obtain the modified hidden representation H'. (ii: KnowPAML). At each k we concatenate o_q^k with h^k to obtain H''. We finally multiply this representation with W_{map} (a trainable matrix) to obtain the final modified hidden representation H'.

The decoder works on this obtained H' to produce the output. We use cross-entropy loss with decoder output and reference output to evaluate generation loss $L^g_{T_{p_i}}$ for dialogue model f_W, which is expressed as:

$$L^g_{T_{p_i}}(f_W) = -\sum logp(u_m | u_{1:m-1}, o_q^k; W) \quad (8)$$

Where, u_m is actual response, $u_{1:m-1}$ is dialogue history, o_q^k is attended representation of triples T and W is weight parameter.

In addition to generation loss, we evaluate the triple representation loss $L^t_{T_{p_i}}$ by using Equation 9 to train the optimal triple representation, following Bordes et al. (2013).

$$h_{emb} = t_{emb} - r_{emb} \quad (9)$$

During meta-training, total loss $L^{total}_{T_{p_i}}$ is calculated as the sum of generation loss and triple representation loss.

$$L^{total}_{T_{p_i}} = L^g_{T_{p_i}}(f_W) + L^t_{T_{p_i}}(f_W) \quad (10)$$

After training one batch of T_{p_i}, the model f_W, parameterized by W, is updated to $W\prime$ using SGD,

$$W'_{p_i} = W - \eta_{inner} \nabla_W L^{total}_{T_{p_i}}(f_W) \quad (11)$$

Where, η_{inner} is inner optimization learning rate and $L^{total}_{T_{p_i}}(f_W)$ is total training loss.

After the parameters are updated, meta-optimization is performed on the unseen dialogues from V_{p_i} set using the updated model $f(W'_{p_i})$ to enhance the model's performance. According Finn et al. (2017), the meta-objective is defined as follows:

$$\min_W \sum_{P_{m_i}} L_{V_{p_i}}(f_{W'_{p_i}}) = \sum_{P_{m_i}} L_{V_{p_i}}(f_{(W - \eta_{outer} \nabla_W L_{V_{p_i}}(f_W))}) \quad (12)$$

Where, $L_{V_{p_i}}(f_{W'_{p_i}})$ is the loss calculated on V_{p_i} set. During this step, we are only considering the dialogue generation loss rather than the total loss.

The initial parameters W are then adjusted using SGD by computing the gradient of average loss, which is the total of $L_{V_{p_i}}(f_{W'_{p_i}})$ obtained at each sampled persona divided by the batch size S. This is expressed as follows:

$$W \leftarrow W - \eta_{outer} \nabla_W \frac{1}{S} \sum_{P_{m_i}} L_{V_{p_i}}(f_{W'_{p_i}}) \quad (13)$$

Where, η_{outer} is the learning rate of outer optimization and $L_{V_{p_i}}(f_{W'_{p_i}})$ is the generation loss calculated on the V_{p_i} set by the updated model parameter $f_{W'_{p_i}}$. This is achieved by the use of second-order partial differentiation.

The validation set P_m^{valid} now validates this meta-training and meta-optimization after every ten iterations. We are dividing the training $T_{p'_i}$ and validating $V_{p'_i}$ set from the P_m^{valid} and performing the same step without altering the original parameters. And then save the best model based on the $L_{V_{p_i}}(W\prime)$. Also model get to know when to terminate training by increasing the count if the loss $L_{V_{p_i}}(W\prime)$ is not the best loss. Algorithm 1 gives an overview of the model.

3.2 Triple Retrieval

We use the ConceptNet, a commonsense knowledge graph, which connects words and phrases with labelled edges and has millions of concepts and edges associated with it. For every dialogue, we take concepts from the dialogue history of length one and associated persona statements to find the ConceptNet neighbours of each concept up to two hops. for example, zero-hop concept C_0 (which is taken from dialogue history and persona statements) is associated with one-hop concept C_1 (all immediate neighbours and all relation between them) and one-hop concept C_1 is associated with two-hop concept C_2. (head concept, relation, tail concept). The top 100 concepts-relations based on weights are then formed into triples (head concept, relation between them, tail concept).

4 Experiments

Our experiments are described in this section, including the dataset, implementation details, baselines, and evaluation metrics.

4.1 Dataset

The PERSONA-CHAT dataset (Zhang et al., 2018), which was also used in the PAML framework (Madotto et al., 2019), was employed for our experiments. The dataset has 1155 different personas in the train data and 100 each in the validation and test data. Each dialogue in this has 4 to 5 persona statements associated with it, and each unique persona has an average of 8.3 unique dialogues.

4.2 Implementation Details

We used the Transformer architecture (Vaswani et al., 2017) which includes six encoder, six decoder layers and four attention heads, just like Madotto et al. (2019), with Glove embedding (Pennington et al., 2014). Here, Transformer's hidden dimension and word embedding dimension are both set at 300. We utilized two different optimizers: SGD for training (inner loop optimizer) and ADAM (Kingma and Ba, 2014) for optimization (outer loop optimizer) with learning rates of 0.01 and 0.0003, respectively and the batch size is set to 16 for both the inner and outer loops.

4.3 Evaluation

We are employing both automatic and human evaluation metrics to evaluate the quality of response compared to baselines.

Automatic Evaluation We evaluate the perplexity (PPL), BLEU score (Papineni et al., 2002), and entailment score (entl), also known as the c score in Madotto et al. (2019). The PPL of the model indicates how well it understands the task on the test set; the lower the perplexity, the better the model understands the language. The BLEU score reflects how close the generated response is to the actual response; usually, a higher BLEU score suggests that the generated response is more comparable to the actual response; however, this cannot be asserted for every scenario. The entl score denotes how much persona information was included in the generated response from persona assertions. If the generated response entails the persona, the score is 1, if the response is independent of the persona, the score is 0, and if the response contradicts the persona information, the score is -1. The higher the entailment score, the more consistent the model is with the persona. This score is calculated using Madotto et al. (2019)'s fine-tuned BERT model, which was trained on persona-based Dialog NLI (Welleck et al., 2018) dataset and has an accuracy of 88.43%.

Human Evaluation We evaluate the Fluency (flcy) to measure the generated response's grammatical

Persona	
I go to the gym 4 days a week. I only drink water. I work in labor and delivery. I am happy being single and alone. I do not want children.	
Context	
Speaker 1:	hello. how are you doing ?
Speaker 2:	hi , i am doing great . how are you ?
Speaker 1:	feeling crabby , but i am like that naturally anyway .
Speaker 2:	oh ok . what do you do for a living ?
Speaker 1:	i am a kennel cleaner at a local animal shelter. and you ?
Speaker 2:	i am a doctor in the labor and deliver unit .
Speaker 1:	sounds very important . you must be a people person .
Speaker 2:	thanks i try to be . i love being along more than it looks like
Speaker 1:	i much prefer to hang with animals than people .
Speaker 2:	what do you do for fun ? i go work out 4 times a week
Speaker 1:	play video games and watch movies . you must be in good shape .
Speaker 2:	i try to be i will not drink soda or even tea , just water
Speaker 1:	that sounds like a healthy lifestyle .
Speaker 2:	it was hard to get use to at first . i use to love soda
Speaker 1:	i still love soda , especially sprite . do you have lots of friends or family ?
Responses	
Gold	i have family and a handful of friends when i am off i keep to myself
Corpus + Persona + Finetuning	i work at the hospital in labor and delivery
Corpus + Finetuning	i do not by the water i am okay and i am okay
PAML + Persona	i work at the gym at work at work
PAML	no it is easy for work
KnowPAML + Persona	i work at the hospital in labor and delivery
AKnowPAML	yes it does . it depends on the woman
KnowPAML	i do not have any , i like living alone .

Table 1: Example of responses generated by the implemented models using 10-shot.

correctness or readability, the Consistency (Con) to measure the persona information included in the response from persona assertions, and the Coherence (Coh) to measure the generated response's relevance with reference to the dialogue history. Three post-graduate-level human experts were asked to rate 200 randomly selected responses from 20 different personas which is generated by the proposed methodology. They were asked to rate consistency on a scale of -1 to 1, with -1 reflecting a contradiction of persona in the generated response, 0 reflecting neutral, and 1 reflecting persona-consistent. And fluency and coherence on a scale of 1 to 3. In fluency, 1 represents poor, 2 represents moderate, and 3 represents excellent in grammatical correctness or readability. And in coherence, 1 for inappropriate responses with context, 2 for moderately coherent responses with context, and 3 for contextually coherent responses.

4.4 Experimental Settings

In our research, we evaluated various training settings in the Transformer model:

PAML: Madotto et al. (2019) proposed this framework, It is a meta-trained model that is tested after fine-tuning the model with dialogues from the same persona.

PAML + Persona: Similar to PAML, except persona statements are included with dialogue history.

Corpus + Finetuning: A model is trained traditionally with dialogue history only as Eq.2 but fine-tuned and tested as PAML.

Experiment	10-Shot			5-Shot		
	PPL	BLEU	Entl	PPL	BLEU	Entl
Corpus + Persona + Finetuning	73.45	0.38	0.09	284.73	0.18	-0.05
Corpus (Varshney et al., 2020) + Finetuning	59.81	0.44	0.06	65.16	0.26	0.03
PAML + Persona	70.55	0.28	0.05	59.4	0.37	0.06
PAML (Madotto et al., 2019)	**41.88**	0.87	0.18	**40.55**	0.87	0.1
KnowPAML + Persona	56.32	0.87	0.11	55.66	0.8	0.06
AKnowPAML	45.78	**1.23**	0.18	48.6	**1.29**	0.08
KnowPAML	48.59	0.76	**0.24**	45.36	0.83	**0.15**

Table 2: Automatic Evaluation Results: When comparing with PAML, KnowPAML and AKnowPAML demonstrate that including concept into the model improves Persona Consistency and BLEU score respectively.

Experiment	10-Shot			5-Shot		
	flcy	Con	Coh	flcy	Con	Coh
Corpus + Persona + Finetuning	2.33	0.12	1.49	2.08	0.07	1.38
Corpus (Varshney et al., 2020) + Finetuning	2.03	0.08	1.53	1.91	0.06	1.46
PAML + Persona	2.36	0.09	1.35	2.11	0.05	1.29
PAML (Madotto et al., 2019)	2.77	0.24	2.17	2.63	0.13	2.06
KnowPAML + Persona	2.71	0.16	2.06	2.56	0.09	1.97
AKnowPAML	**2.89**	0.25	**2.33**	**2.74**	0.14	**2.23**
KnowPAML	2.83	**0.33**	2.21	2.64	**0.18**	2.13

Table 3: Human Evaluation Results: When comparing with PAML, KnowPAML and AKnowPAML demonstrate that including concept into the model improves Consistency (Con) and Coherence (Coh) respectively.

Corpus +Persona + Finetuning: Similar to the last one, except persona statements are included with dialogue history.

KnowPAML: We include knowledge in the form of triples with a multi-hop attention mechanism into the PAML using (setting (ii) section 3.1)

KnowPAML + Persona: Similar to KnowPAML, except persona statements are included with dialogue history.

AKnowPAML: This is similar to KnowPAML, but it differs in how attended triples are integrated into encoder outputs (setting (i) section 3.1).

We created dialogue history by appending all past utterances in the dialogue, with the length dictated by the number of turns that occurred, so there is no predetermined length. It will change with the turns, and there are no limits to the number of turns. We showed the results of 5 and 10 shots, where the response from the dialogue context and associated persona is generated after finetuning on 5 and 10 dialogue with the same persona, respectively.

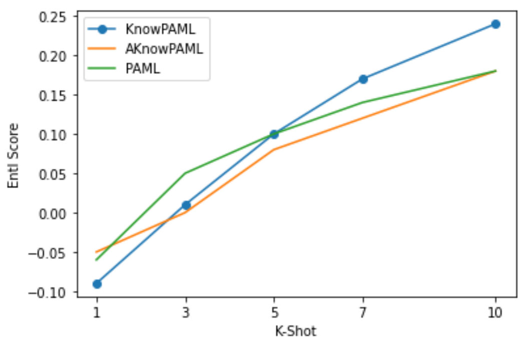


Figure 2: Entl Score vs K-Shot results of KnowPAML, AKnowPAML and PAML frameworks.

5 Result and Discussion

The results of the automatic and human evaluation for the 5-shot and 10-shot settings are shown in Table 2 and Table 3 respectively. KnowPAML was found to be most consistent to the persona in both automatic and human evaluation when compared to other systems. This implies that by incorporating persona and dialogue history knowledge into the existing framework, the system generates responses with a greater amount of persona information. AKnowPAML, on the other hand, performed better in terms of BLEU, fluency (flcy), and coherence (Coh). Perplexity (PPL), however, is higher in both instances, demonstrating a negative correlation between PPL and human likeness (Doğruöz and Skantze, 2021). KnowPAML outperformed PAML by 33.3% in 10-shots and 50% in 5-shots in terms of Entl score. In terms of consistency score (human evaluation), KnowPAML outperformed PAML by 37.5% in 10-shots and 38.5% in 5-shots. In the case of AKnowPAML, the Entl score is similar to the PAML score, while the BLEU score outperforms by 41.4% in 10-shots and 48.3% in 5-shots. Fluency and coherence are equivalent in PMAL, AKnowPAML, and KnowPAML in terms of human evaluation. The Entl score versus K-Shot results of KnowPAML, AKnowPAML, and PAML frameworks are shown in Figure 2, where K is 1, 3, 5, 7, and 10. Shots refer to the number of dialogues, such as 1-shot means one dialogue, 3-shot means three dialogue, that is used to fine-tune. Figure 2 shows that KnowPAML behaves more linearly compare to PAML and AKnowPAML. Entl score of KnowPAML intersects with PAML's score at 5-shot and then it keeps improving over PAML.

Persona:
i have got two more years in college i study law i want to be successful i am a student i have no siblings
Context:
hello , how are you this evening ?
Responses:
Corpus + Persona + Finetuning: i am well just studying
PAML + Persona: i am doing good and you ?
PAML: i am well just studying
KnowPAML + Persona: i am well just studying for class
AKnowPAML: fine . are you from one of my law classes ?
KnowPAML: fine . are you from one of my law classes ?

Table 4: Comparison of response generated by the PAML and KnowPAML using 10-shots.

5.1 Analysis

Table 1 shows an example of generated responses from different models using 10-shot fine tuning on held-out persona. As the PERSONA-CHAT dataset is an open domain conversation, the flow of the conversation can go in any direction, and the speaker generally talks in consonance with their persona. For new persona (not present in training data) it is often difficult to generate meaningful persona grounded responses that are also contextually relevant.

In Table 1, speaker 1 wants to know whether speaker 2 has a lot of friends or family; the response given by the KnowPAML framework is correct and consistent with the persona and context. In contrast, the response generated by the other framework is consistent with the persona but not with the context. The response from the PAML system in this case is inconsistent with both the persona and the context. In Table 4, The speaker asks, "How are you this evening?". All frameworks generate correct response given the context, however, our model generates a more engaging response that is consistent with one of the persona statements.

It has also been observed that responses generated by KnowPAML have an advantage in generating more persona consistent responses compared to AknowPAML, whereas AknowPAML attempts to generate depending on the real response, which is why the BLEU score is high. However, in terms of fluency, all are comparable, demonstrating that BLEU does not correlate with human judgment, as underlined by Liu et al. (2016) in their research of "How Not to Evaluate Your Dialogue System".

6 Conclusion

In this paper, we proposed the KnowPAML framework, which uses a multi-hop attention mechanism to absorb concepts in the form of triples from the ConceptNet knowledge graph in a Meta-learning setting. Our PERSONA-CHAT experiment demonstrates the advantage of using KnowPAML over previous frameworks in terms of persona-adaptability, resulting in more persona-consistent responses. The analysis of the generated responses reveals that the knowledge added model can successfully aid in persona adaptability, consistent response generation, and conversation flow. Although fluency and coherence are comparable to those of others, they can be improved further in the future by using a pre-trained language model.

Acknowledgement

The research reported in this paper is an outcome of the project “Autonomous Goal-Oriented and Knowledge-Driven Neural Conversational Agents", sponsored by Accenture LLP. Asif Ekbal acknowledges Visvesvaraya Young Faculty Research Fellowship (YFRF).

References

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

Siqi Bao, Huang He, Fan Wang, Rongzhong Lian, and Hua Wu. 2019. Know more about each other: Evolving dialogue strategy via compound assessment. *arXiv preprint arXiv:1906.00549*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

A Seza Doğruöz and Gabriel Skantze. 2021. How “open” are the conversations with open-domain chatbots? a proposal for speech event based evaluation. In *The 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL’22)*.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2018. A knowledge-grounded neural conversation model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Jiatao Gu, Yong Wang, Yun Chen, Kyunghyun Cho, and Victor OK Li. 2018. Meta-learning for low-resource neural machine translation. *arXiv preprint arXiv:1808.08437*.

Hiroaki Hayashi, Zecong Hu, Chenyan Xiong, and Graham Neubig. 2020. Latent relation language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7911–7918.

Yi Huang, Junlan Feng, Shuo Ma, Xiaoyu Du, and Xiaoting Wu. 2020. Towards low-resource semi-supervised dialogue generation with meta-learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4123–4128.

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

Jing Yang Lee, Kong Aik Lee, and Woon Seng Gan. 2021. Generating personalized dialogue via multi-task meta-learning. *arXiv preprint arXiv:2108.03377*.

Margaret Li, Stephen Roller, Ilia Kulikov, Sean Welleck, Y-Lan Boureau, Kyunghyun Cho, and Jason Weston. 2019. Don’t say that! making inconsistent dialogue unlikely with unlikelihood training. *arXiv preprint arXiv:1911.03860*.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

Yinong Long, Jianan Wang, Zhen Xu, Zongsheng Wang, Baoxun Wang, and Zhuoran Wang. 2017. A knowledge enhanced generative conversational service agent. In *Proceedings of the 6th Dialog System Technology Challenges (DSTC6) Workshop*.

Andrea Madotto, Zhaojiang Lin, Chien-Sheng Wu, and Pascale Fung. 2019. Personalizing dialogue agents via meta-learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5454–5459.

Bodhisattwa Prasad Majumder, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Julian McAuley. 2020. Like hiking? you probably enjoy nature: Persona-grounded dialog with commonsense expansions. *arXiv preprint arXiv:2010.03205*.

Pierre-Emmanuel Mazaré, Samuel Humeau, Martin Raison, and Antoine Bordes. 2018. Training millions of personalized dialogue agents. *arXiv preprint arXiv:1809.01984*.

Fei Mi, Minlie Huang, Jiyong Zhang, and Boi Faltings. 2019. Meta-learning for low-resource natural language generation in task-oriented dialogue systems. *arXiv preprint arXiv:1905.05644*.

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.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Kun Qian and Zhou Yu. 2019. Domain adaptive dialog generation via meta learning. *arXiv preprint arXiv:1906.03520*.

Haoyu Song, Wei-Nan Zhang, Jingwen Hu, and Ting Liu. 2020. Generating persona consistent dialogues by exploiting natural language inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8878–8885.

Yiping Song, Zequn Liu, Wei Bi, Rui Yan, and Ming Zhang. 2019. Learning to customize model structures for few-shot dialogue generation tasks. *arXiv preprint arXiv:1910.14326*.

R Speer, J Chin, and C ConceptNet Havasi. 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (December 2016)*, pages 4444–4451.

Nam Khanh Tran and Claudia Niedereée. 2018. Multihop attention networks for question answer matching. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 325–334.

Deeksha Varshney, Asif Ekbal, Ganesh Prasad Nagaraja, Mrigank Tiwari, Abhijith Athreya Mysore Gopinath, and Pushpak Bhattacharyya. 2020. Natural language generation using transformer network in an open-domain setting. In *International Conference on Applications of Natural Language to Information Systems*, pages 82–93. Springer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Pavlos Vougiouklis, Jonathon Hare, and Elena Simperl. 2016. A neural network approach for knowledge-driven response generation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3370–3380.

Sean Welleck, Jason Weston, Arthur Szlam, and Kyunghyun Cho. 2018. Dialogue natural language inference. *arXiv preprint arXiv:1811.00671*.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. Transfertransfo: A transfer learning approach for neural network based conversational agents. *arXiv preprint arXiv:1901.08149*.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie Sun, and Xiaolong Wang. 2017. Incorporating loose-structured knowledge into conversation modeling via recall-gate lstm. In *2017 international joint conference on neural networks (IJCNN)*, pages 3506–3513. IEEE.

Houyu Zhang, Zhenghao Liu, Chenyan Xiong, and Zhiyuan Liu. 2019. Grounded conversation generation as guided traverses in commonsense knowledge graphs. *arXiv preprint arXiv:1911.02707*.

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? *arXiv preprint arXiv:1801.07243*.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018a. Commonsense knowledge aware conversation generation with graph attention. In *IJCAI*, pages 4623–4629.

Kangyan Zhou, Shrimai Prabhumoye, and Alan W Black. 2018b. A dataset for document grounded conversations. *arXiv preprint arXiv:1809.07358*.

Wenya Zhu, Kaixiang Mo, Yu Zhang, Zhangbin Zhu, Xuezheng Peng, and Qiang Yang. 2017. Flexible end-to-end dialogue system for knowledge grounded conversation. *arXiv preprint arXiv:1709.04264*.

There is No Big Brother or Small Brother: Knowledge Infusion in Language Models for Link Prediction and Question Answering

Ankush Agarwal[1*], **Sakharam Gawade**[1*],
Sachin Channabasavarajendra[2], **Pushpak Bhattacharyya**[1]
[1]IIT Bombay,
[2]Honeywell Technology Solutions Pvt Ltd
`{ankushagrawal, sakharamg, pb}@cse.iitb.ac.in,`
`sachin.channabasavarajendra@honeywell.com`

Abstract

The integration of knowledge graphs with deep learning is thriving in improving the performance of various natural language processing (NLP) tasks. In this paper, we focus on knowledge-infused link prediction and question answering using language models, T5, and BLOOM across three domains: Aviation, Movie, and Web. In this context, we infuse knowledge in large and small language models and study their performance, and find the performance to be similar. For the link prediction task on the Aviation Knowledge Graph, we obtain a 0.2 hits@1 score using T5-small, T5-base, T5-large, and BLOOM. Using template-based scripts, we create a set of 1 million synthetic factoid QA pairs in the aviation domain from National Transportation Safety Board (NTSB) reports. On our curated QA pairs, the three models of T5 achieve a 0.7 hits@1 score. We validate our findings with the paired student t-test and Cohen's kappa scores. For link prediction on Aviation Knowledge Graph using T5-small and T5-large, we obtain a Cohen's kappa score of 0.76, showing substantial agreement between the models. Thus, we infer that small language models perform similar to large language models with the infusion of knowledge.

1 Introduction

A large number of pre-trained language models (LMs) are used for downstream tasks, such as Question Answering (QA). Generally, these language models are trained on generic domain data, such as Web data and News Forums. Recently, LMs are used for downstream tasks in domain-specific fields, namely, healthcare (Michalopoulos et al., 2021), radiology (Kale et al., 2022), and aviation (Agarwal et al., 2022). For tasks such as Information Extraction (IE) and Question Answering (QA), Knowledge Graphs (KGs) are used as a source of external knowledge to boost the performance of models. To a great extent, researchers focus on the synergy of Knowledge Graph and Deep Learning (Miller et al., 2016a; Saxena et al., 2020, 2022). With the increase in data, it is observed that larger models are preferred for different tasks across various domains.

The Large Language Models (LLMs) are preferred to obtain better results than small or non-pre-trained models as they have a vast number of parameters and have been trained on a large amount of data. But, the larger model increases the need for computation power and training time. In this paper, we show that small and large models perform likewise with the infusion of knowledge. We can use non-pre-trained models for different tasks across domains that require less computation power and time and still attain the same performance as pre-trained models.

We validate our hypothesis with the LLMs, *i.e.*, T5 (Raffel et al., 2020) & BLOOM[1]. We perform two tasks: a) Link Prediction, and b) Question Answering on different datasets: a) Aviation Knowledge Graph (AviationKG) (Agarwal et al., 2022), and Aviation QA pairs (section 4.4), b) Movie Knowledge Base (MovieKB) & MetaQA (a set of QA pairs), both present in the MetaQA dataset (Zhang et al., 2018), and c) Complex Web Questions (CWQ) (Talmor and Berant, 2018), which uses subsets of Freebase (Chah, 2017). We perform hypothesis testing to validate our hypothesis. We use paired Student T-test and attempt to reject our hypothesis that models have a negligible difference in performance. But, we were not able to repudiate our hypothesis. To strengthen our findings, we use Cohen's kappa measure and show significant agreement between models.

Our **contributions** are as follows:

[*]Equal contribution

[1]`https://huggingface.co/bigscience/bloom`

1. We create a synthetic dataset, AviationQA [2], a set of 1 million factoid QA pairs from 12,000 National Transportation Safety Board (NTSB) reports using templates explained in section 4.4. These QA pairs contain questions such that answers to them are entities occurring in the AviationKG (Agarwal et al., 2022). AviationQA will be helpful to researchers in finding insights into aircraft accidents and their prevention.

2. We show that the size of a language model is inconsequential when knowledge is infused from the knowledge graphs. With AviationKG, we obtain 0.22, 0.23, and 0.23 hits@1 scores for link prediction using T5-small, T5-base, and T5-large, respectively. On AviationQA, we get a 0.70 hits@1 score on the three sizes of the T5 model. We validate our hypothesis with paired student t-test, and Cohen's kappa explained in section 6. We obtain a substantial Cohen's kappa score of 0.76 for link prediction on AviationKG using T5-small and T5-large. For Question Answering using T5-small and T5-large, we get a Cohen's kappa score of 0.53 on the MetaQA dataset. Hence, we provide evidence that we can substitute larger models with smaller ones and achieve the same performance with less computational cost and power.

2 Motivation

As stated earlier, in Section 1, LMs are trained on generic datasets. So, knowledge from different sources, *i.e.*, KGs, are used to perform downstream tasks in specific domain areas. LLMs infused with knowledge are required to perform such tasks, namely, QA and link prediction, which increases the need for computation power and time. We show that computational resources can be saved by using smaller language models for tasks.

It is rare to obtain datasets related to the aviation domain, which is in increased demand. We scrape NTSB reports from NTSB's website [3] and create QA pairs that can be used by the aviation industry and researchers for Information Retrieval (IR) and QA purposes. The created dataset will help find insights into aircraft accidents and develop solutions to prevent accidents.

[2]https://github.com/ankush9812/Aviation-Question-Answer-Pairs

[3]https://www.ntsb.gov/Pages/AviationQuery.aspx

3 Background & Related Work

A Knowledge Graph is a collection of entities and relations represented in the form of triplets (subject, relation, object). Querying KG in Natural Language (NL) is a long-standing work. Early work focused on rule-based and pattern-based systems (Affolter et al., 2019). Recently, the work is shifted to seq2seq architecture (Zhong et al., 2017) and pre-trained models with the advent of neural networks. Querying KGs remains a challenge because of the conversion of NL to the graph query language, namely, SPARQL, Cypher, etc.

With the value increase of knowledge in the world, the popularity of the KG has escalated. Researchers are keenly interested in the synergy of knowledge graphs and deep learning. Several methods are exploited considering synergy: a) Integrating triplets of KG into the neural network (Liu et al., 2020; Saxena et al., 2022), b) Computing the relevance of entity and relations in a KG using a neural network (Sun et al., 2019; Yasunaga et al., 2021).

Deep Learning models use representations of entities and relations to integrate triplets of KG. Knowledge Graph Embeddings are widely used to obtain representations (Dai et al., 2020). The KG embedding models are trained on link prediction over triplets to obtain representations (Wang et al., 2021). Recent work has focused on using fine-tuned language models over KGE models for link prediction to reduce the number of parameters required to obtain the representations (Saxena et al., 2022).

LMs and KGs are extensively used to improve task-specific performance. Still, no study has been done to understand the characteristics of a language model during the synergy of KG and DL. In this paper, we observe the behavior of language models after knowledge infusion with different domain datasets.

4 Methodology and Experimental Design

This section presents our approach (flow diagram in figure 1), discusses the experiment datasets, creation of AviationQA, describes the model configurations, and explains the evaluation technique.

4.1 Approach

We observe the performance of small and large language models with the infusion of knowledge for link prediction and QA. Experiments are performed with the following models (detailed in section 4.6): a) T5-small non-pre-trained, b) T5-base pre-trained, c) T5-large pre-trained, and SOTA d) BLOOM 1b7. We make use of different domain datasets for our approach, explained in section 4.2. Figure 1 demonstrates link prediction and question answering on the data after pre-processing.

We inject knowledge into the LMs. The knowledge is injected by the process of fine-tuning the pre-trained LM. Fine-tuning requires a learning objective and training data. In our case, the training data is triplets from the KG (table 1), and the learning objective is triple completion. Triple completion involves getting tail entity given head entity and relation. Triple completion is also called link prediction. Thus, the LM absorbs the knowledge. The link prediction results with triplets are shown in table 3.

After fine-tuning on triplets for link prediction, the language model learns representations of entities and relations. The checkpoint with the best result on link prediction is used for the question-answering task. We again fine-tune the selected checkpoint with QA pairs (table 2) and obtain the QA results shown in table 4.

4.2 Experiment Data

We are using three datasets: a) Aviation Knowledge Graph (AviationKG) (Agarwal et al., 2022) & Aviation QA pairs (section 4.4), b) MetaQA (Zhang et al., 2018), which consists of a KB constructed from WikiMovies dataset (Miller et al., 2016b) and question-answer pairs, and c) Complex Web Questions (CWQ) (Talmor and Berant, 2018), which uses subsets of Freebase (Chah, 2017). The statistic of these datasets is shown in table 1 & 2. We chose these datasets because they belong to different domains and vary in size.

MetaQA KB & AviationKG are from the movie and aviation domains, respectively, which is useful to represent the diversity of datasets and validate our hypothesis. CWQ is based on Freebase, a huge KG, which is crowd-sourced. We require a knowledge base and the corresponding QA pairs for our experimentation, described in section 4.5. MetaQA and CWQ are openly available datasets. But, there is no available QA pairs dataset for the aviation domain. We create a set of QA pairs in the aviation domain and contribute to the research community, detailed in section 4.4. The datasets used in the paper are pre-processed and split before running experiments, as explained in section 4.3 and 4.5.

Dataset	Train	Validation	Test
AviationKG	173,372	10,000	10,000
MovieKB	249,482	10,000	10,000
CWQ	27,590,648	10,000	10,000

Table 1: Statistics of triplets (subject, relation, object) for three knowledge bases: AviationKG (Agarwal et al., 2022), MetaKB (Zhang et al., 2018), and Complex Web Question (CWQ) (Talmor and Berant, 2018). Subsets of Freebase (Chah, 2017) are used for CWQ.

Dataset	Train	Validation	Test
AviationQA	367,304	10,000	10,000
MetaQA	184,230	10,000	10,000
CWQ	61,619	3,519	3,531

Table 2: Statistics of Question Answer pairs from three domains: Aviation, Movie, and Web. For MetaQA, we use 1-hop questions. For more details, refer to section 4.5.

4.3 Data Pre-processing

We make use of KG and QA pairs (section 4.2) from 3 domains, Aviation, Movie, and General domain. These datasets are cleaned and structured for our experiments. For the link prediction task, the dataset is created similar to Saxena et al. (2022), described below:
predict head: subject | relation | object
predict tail: object | relation | subject
The triplets {subject, relation, object} are extracted from the AviationKG, MovieKB, and Freebase individually.

All these knowledge bases are associated with the corresponding QA pairs. As explained in section 4.4, we construct the AviationQA pairs and use MetaQA 1-hop and CWQ for question answering. For QA fine-tuning, the dataset is in the given format:
predict answer: question | answer.
E.g., **predict answer**: What is the capital of India? | New Delhi.
Multiple answers exist for a question in AviationQA, MetaQA, and CWQ. These collective instances are separated as individual QA pairs.

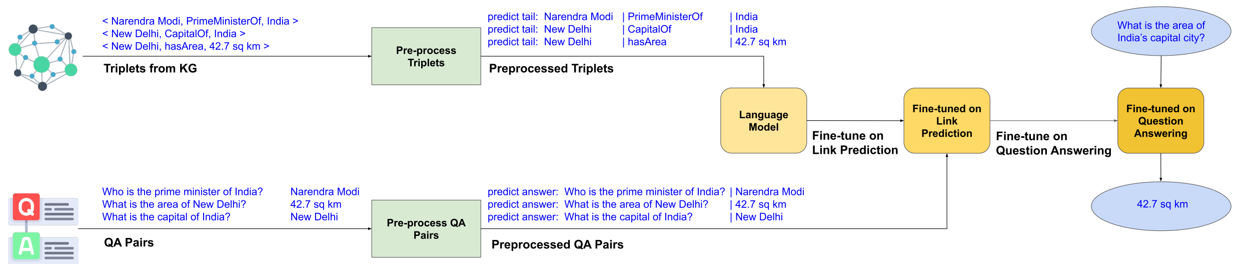

< Narendra Modi, PrimeMinisterOf, India >
< New Delhi, CapitalOf, India >
< New Delhi, hasArea, 42.7 sq km >
Triplets from KG
Pre-process Triplets
predict tail: Narendra Modi | PrimeMinisterOf | India
predict tail: New Delhi | CapitalOf | India
predict tail: New Delhi | hasArea | 42.7 sq km
Preprocessed Triplets
Language Model
Fine-tune on Link Prediction
Fine-tuned on Link Prediction
Fine-tune on Question Answering
Fine-tuned on Question Answering
What is the area of India's capital city?
42.7 sq km
Who is the prime minister of India? Narendra Modi
What is the area of New Delhi? 42.7 sq km
What is the capital of India? New Delhi
QA Pairs
Pre-process QA Pairs
predict answer: Who is the prime minister of India? | Narendra Modi
predict answer: What is the area of New Delhi? | 42.7 sq km
predict answer: What is the capital of India? | New Delhi
Preprocessed QA Pairs

Figure 1: Flow diagram of the approach adopted in our paper. The model is first fine-tuned on KG triplets for Link Prediction. Next, the fine-tuned model is again fine-tuned on question answering. Because of the link-prediction task, the model learns KG completion and can answer multi-hop questions. E.g., If the model knows India's capital is New Delhi and New Delhi's area size, then the model should predict the area of India's capital correctly without explicitly mentioning New Delhi in the question

E.g., What countries did Narendra Modi visit in the year 2021? Answers: United States, Italy. Every QA pair is segregated in the current layout: a) What countries did Narendra Modi visit in the year 2021? | United States. b) What countries did Narendra Modi visit in the year 2021? | Italy.

With small KGs, *i.e.*, AviationKG, and MovieKB, triplet samples are added during QA fine-tuning to avoid overfitting. The added triplets are in the same format as mentioned for the link prediction task. The pre-processing of triplets and QA pairs is shown in figure 1.

4.4 Creation of AviationQA

We web scrape the National Transportation Safety Board (NTSB) website and download 12k reports from 2009-2022. A set of 90 question templates is prepared using the common structure of documents in the format:

- Where did the accident [ ] take place?
- What is the model/series of the aircraft bearing accident number [ ]?
- Was there fire on the aircraft of the accident number [ ]?

The template of questions is created, and answers to those questions are extracted from every NTSB report. Because every report is associated with an accident number, we place [ ] in the template to indicate which report the question pertains to, e.g., CHI07LA273, LAX07LA148. NTSB reports are semi-structured, containing unstructured data in paragraphs and structured data in tabular format. We extract answers from each report w.r.t the template using the regular expression method. Later, QA pairs are scrutinized. As some reports' structure varies, different scripts are written to fetch answers for those reports.

We successfully created 1 million factoid QA pairs in the aviation domain using the template-based method. The dataset will contribute to research and development in the aviation industry.

4.5 Dataset Description

After pre-processing the data (section 4.3), we split it to train, validate, and test for link prediction and question answering. Table 1 shows the split of triplets from AviationKG, MovieKB, and subsets of Freebase. CWQ uses subsets of Freebase, which is of size 27 million. AviationKG and MovieKB are domain-specific datasets of sizes 170k and 250k. Valid and test splits are equal in size to 10k each.

Our motive for considering different sizes and domain datasets is to strengthen our intuition that the performance of varying size models remains the same with an infusion of knowledge in language models. Table 3 shows the correctness of our intuition with the link prediction task.

Table 2 shows the split of QA pairs for question-answering. We use 387,304 instances for AviationQA from 1 million QA pairs (section 4.4). The scrutinization is based on reports used to create AviationKG (Agarwal et al., 2022) from 1962 to 2015. We use QA pairs that have information available in the AviationKG. Moreover, we ensured that an answer to a question is an entity in the AviationKG.

For comparison between the movie and the aviation data, the split of valid and test set is the same in both, *i.e.*, 10k. CWQ dataset is smaller than AviationQA and MetaQA, so we use the same validation and test split, as mentioned in Saxena et al. (2022).

4.6 Model Configuration

In this paper, we are using four models: T5-small non-pretrained (60 million parameters), T5-base pre-trained (220 million parameters), T5-large pre-trained (770 million parameters), and BLOOM (1.72 billion parameters). These models are considered to validate our statement that with the injection of knowledge, small and large model performs the same. Both tasks, link prediction and question answering, are performed using these models. The T5 model is considered in our experiment as it is trained to perform multiple downstream tasks, i.e., translation, classification, and question answering. We use BLOOM as it is similar to the SOTA model GPT-3 (Brown et al., 2020), which has outperformed other language models on tasks such as QA and summarization.

4.7 Evaluation Technique

We evaluate the performance of our models using the hits@1 score for link prediction and question answering. Table 3 and 4 show the hits@1 score for link prediction and question answering, respectively, on different datasets. We choose the hits@1 score for evaluation as it is more precise than other hits@k scores. If the first predicted value matches the actual answer, then the score is 1; otherwise, 0. We are using the hits@1 metric and not other metrics such as BLEU score (Papineni et al., 2002) and semantic similarity (Miller and Charles, 1991) to validate the correctness of our hypothesis (introduced in section 1). BLEU score is generally used for comparing sentences, whereas, for link prediction and QA tasks, the answer is a compound noun, *i.e.*, an entity in the knowledge graph. Since the entities are ranked for tasks, the hits@1 score is the best metric. As the answers to link prediction and QA are entities of KG, the semantic similarity would not be able to distinguish between 2 different entities with semantically the same meaning. After considering all drawbacks of other metrics, we adapted the hits@1 score for the evaluation.

5 Results and Analysis

This section analyzes the performance of two models: T5 and BLOOM. Table 3 & 4 show the hits@1 score for link prediction and QA tasks, respectively. With table 3, we can clearly observe that the hits@1 score for three variations of the T5 model & BLOOM is proximate for three different datasets (section 4.5). The three T5 models score 0.22 & 0.23 hits@1 for link prediction on AviationKG. Similarly, scores with MetaKB and CWQ have very less differences among models. LMs on MetaKB perform poorly for link prediction compared to other datasets; 0.02 & 0.03 are the hits@1 scores on the T5 model & BLOOM. The reason is the extensiveness of triplets in the MetaKB and the presence of noise in the dataset. We chose MetaKB to have a diversity of datasets and justify our claim (explained in section 1).

The main observation with the link prediction task is that the T5-small non-pre-trained model performs alike to pre-trained models. The T5-base with 220 million parameters shows results like T5-large & BLOOM, which comprises 770 million & 1.7 billion parameters, respectively. Link prediction results (in table 3) infers our claim that small and large models perform the same with the infusion of knowledge.

To support our claim, we also performed QA with the same set of models as used for the link prediction task. With the AviationQA dataset, we achieved 0.7 hits@1 scores on T5-small, T5-base, and T5-large. LLMs such as T5-large & BLOOM are expected to perform better for QA than small models as they are trained with a large amount of data and vice-versa, T5-small non-pre-trained, and T5-base are expected to perform direly. But, we observe that the performance of all three T5 models

Model	AviationKG	MetaKB	CWQ
T5-small	0.2258	0.0257	0.2153
T5-base	0.2387	0.0286	0.2273
T5-large	0.2323	0.0301	0.2207
BLOOM 1b7	0.2163	0.0365	0.2155

Table 3: Link Prediction results on three knowledge bases: Aviation Knowledge Graph (KG) (Agarwal et al., 2022), Meta Knowledge Base (Zhang et al., 2018), and subsets of Freebase (Chah, 2017) for Complex Web Questions (CWQ) (Talmor and Berant, 2018).

Model	AviationQA	MetaQA	CWQ
T5-small	0.7031	0.2144	0.2225
T5-base	0.7093	0.2158	0.2736
T5-large	0.7013	0.2371	0.2632
BLOOM 1b7	0.5507	0.2386	0.1517

Table 4: Question Answering (QA) results in three QA datasets: AviationQA (4.4), MetaQA (Zhang et al., 2018), and Complex Web Questions (CWQ) (Talmor and Berant, 2018).

Hypothesis Testing	AviationKG			MetaQA		
	T5-small T5-large	T5-base T5-large	T5-large Bloom	T5-small T5-large	T5-base T5-large	T5-large Bloom
Paired Student T-test	Cannot Reject	Cannot Reject	Cannot Reject	Cannot Reject	Cannot Reject	Cannot Reject
Cohen's kappa Score	0.76	0.75	0.68	0.49	0.53	0.33
Agreement (%)	91.77	91.36	89.16	82.50	83.62	75.73

Table 5: Hypothesis Testing on link prediction with 'AviationKG' and question-answering with 'MetaQA' datasets. We choose two measures for the test: a) paired Student T-test (Hsu and Lachenbruch, 2014), and b) Cohen's kappa Score (Cohen, 1968), to prove our hypothesis- after injection of knowledge, small and large models perform the same. Student T-test with 0.1 significance value is done on 2000 instances of the test set selected randomly, and our hypothesis is not rejected 7 out of 10 times. We use the entire test set of 10,000 instances for the kappa score. Cohen's kappa scores on link prediction for AviationKG are between 0.6 and 0.8, and on question-answering for MetaQA, between 0.4 and 0.6. With these scores, we are able to prove that our claim is correct.

is the same for QA with the AviationQA dataset. Similarly, we observe that MetaQA achieves 0.2 hits@1 scores for non-pre-trained T5, pre-trained T5-base, T5-large, and BLOOM.

Through our experiments, we have shown how different model sizes perform on QA after infusion of knowledge using link prediction. Pre-trained and non-pre-trained models of different sizes have shown similar results on different domain datasets for link prediction and QA tasks. This contribution to the research community is pivotal as high accuracy can be achieved efficiently with less computation power, time, and cost.

The source code for our paper is publicly available on GitHub[4].

6 Hypothesis Testing

We attempt to contradict our hypothesis (1) that the difference in scores for the two models is negligible. We choose paired student t-test (Hsu and Lachenbruch, 2014) to refute our hypothesis. In our testing, the significance level (p-value) is 0.1, and the sample size is 20% of the test set selected randomly. In comparing the pair of models (section 4.6), we predicted T5-large to perform better than T5-base & T5-small and Bloom to perform better than all three models of T5 because of its large size. But, 7 out of 10 times student t-test was unable to reject our hypothesis, and the significance level among the pair of models was greater than 0.1. Table 5 clearly shows the paired student t-test on AviationKG (table 1) and MetaQA (table 2) for different pairs of models, and the result is the same, our hypothesis cannot be rejected.

After not being able to reject the hypothesis, our next step was to strengthen it, so, we calculate Cohen's kappa (Cohen, 1968) score of the pair of models with different datasets (table 1 & 2). We consider a pair of models as two annotators and the hits@1 score corresponding to each sample in the test set as their annotations. Since our evaluation technique (section 4.7) uses hits@1 score and the score is binary for each sample, Cohen's kappa score is used to measure the reliability between the two models. The kappa score is calculated for all instances of the test set. Table 5 shows the Cohen's kappa score and % agreement for AviationKG and MetaQA datasets between pair of models. For link prediction on AviationKG, the kappa score is between 0.6 and 0.8, and agreement is near 90%. These results clearly denote the substantiality of our claim with high scores. We extend the test for question-answering with MetaQA. The pair of T5 models score 0.4-0.6, denoting moderate agreement as more than 80% of agreement. T5-large and Bloom pair scores 0.33 with 75.7% agreement, which is fair.

Thus, the testing supports our hypothesis, and we prove that the level of performance of different models with the infusion of knowledge remains the same.

7 Conclusion and Future Work

We have successfully created a million factoid QA pairs from the NTSB aircraft accident reports. The QA pairs are used in our experiments with AviationKG. We have validated our claim that with the infusion of knowledge to language models, the per-

[4]https://github.com/ankush9812/Knowledge-Infusion-in-LM-for-QA

formance of the small language model is similar to the large language model. We substantiate with different language models and a diversity of datasets. Our investigation will benefit researchers in selecting the appropriate language model when working with knowledge and save computation power and time.

The future line of work is to investigate the performance of models with incomplete and noisy knowledge graphs and study the extent to which the models can outright the domain knowledge.

Acknowledgements

This research is supported by the Science and Education Research Board (SERB), Ministry of Education, India, under the Imprint-2 project. We thank our Industry partner, Honeywell Technology Solutions Pvt Ltd, who provided insight and expertise that greatly assisted this research.

References

Katrin Affolter, Kurt Stockinger, and Abraham Bernstein. 2019. A comparative survey of recent natural language interfaces for databases. *The VLDB Journal*, 28(5):793–819.

Ankush Agarwal, Raj Gite, Shreya Laddha, Pushpak Bhattacharyya, Satyanarayan Kar, Asif Ekbal, Prabhjit Thind, Rajesh Zele, and Ravi Shankar. 2022. Knowledge graph–deep learning: A case study in question answering in aviation safety domain. *arXiv preprint arXiv:2205.15952*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Niel Chah. 2017. Freebase-triples: A methodology for processing the freebase data dumps. *arXiv preprint arXiv:1712.08707*.

Jacob Cohen. 1968. Weighted kappa: nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213.

Yuanfei Dai, Shiping Wang, Neal N. Xiong, and Wenzhong Guo. 2020. A survey on knowledge graph embedding: Approaches, applications and benchmarks. *Electronics*, 9(5).

Henry Hsu and Peter A Lachenbruch. 2014. Paired t test. *Wiley StatsRef: statistics reference online*.

Kaveri Kale, Pushpak Bhattacharyya, Aditya Shetty, Milind Gune, Kush Shrivastava, Rustom Lawyer, and Spriha Biswas. 2022. Knowledge graph construction and its application in automatic radiology report generation from radiologist's dictation.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-bert: Enabling language representation with knowledge graph. In *AAAI*.

George Michalopoulos, Yuanxin Wang, Hussam Kaka, Helen Chen, and Alexander Wong. 2021. UmlsBERT: Clinical domain knowledge augmentation of contextual embeddings using the Unified Medical Language System Metathesaurus. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1744–1753, Online. Association for Computational Linguistics.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016a. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409, Austin, Texas. Association for Computational Linguistics.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016b. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409, Austin, Texas. Association for Computational Linguistics.

George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28.

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.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

Apoorv Saxena, Adrian Kochsiek, and Rainer Gemulla. 2022. Sequence-to-sequence knowledge graph completion and question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2814–2828.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4498–4507, Online. Association for Computational Linguistics.

Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 2019. PullNet: Open domain question answering with iterative retrieval on knowledge bases and 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 2380–2390, Hong Kong, China. Association for Computational Linguistics.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. 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 641–651, New Orleans, Louisiana. Association for Computational Linguistics.

Meihong Wang, Linling Qiu, and Xiaoli Wang. 2021. A survey on knowledge graph embeddings for link prediction. *Symmetry*, 13(3).

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546, Online. Association for Computational Linguistics.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Thirty-second AAAI conference on artificial intelligence*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

A Appendix

A.1 Examples of AviationQA

Below, we mention some examples from our created Aviation question-answering dataset (section 4.4):

- **Q:** Which seat was occupied by the pilot responsible for accident no. CEN18LA272?
  A: Left
- **Q:** Are there other Aircraft Rating(s) for the pilot of accident no. GAA18CA489?
  A: None
- **Q:** What is the make of the aircraft bearing accident no. CEN18LA272?
  A: Cessna
- **Q:** What is the category of the aircraft involved in accident no. GAA18CA489?
  A: Gyroplane
- **Q:** What is the Airworthiness Certificate of accident no. GAA18CA297?
  A: Normal

Efficient Joint Learning for Clinical Named Entity Recognition and Relation Extraction Using Fourier Networks: A Use Case in Adverse Drug Events

Anthony Yazdani, Dimitrios Proios, Hossein Rouhizadeh, Douglas Teodoro
University of Geneva, Faculty of medicine,
Department of radiology and medical informatics, Data science for digital health
firstname.lastname@unige.ch

Abstract

Current approaches for clinical information extraction are inefficient in terms of computational costs and memory consumption, hindering their application to process large-scale electronic health records (EHRs). We propose an efficient end-to-end model, the Joint-NER-RE-Fourier (JNRF), to jointly learn the tasks of named entity recognition and relation extraction for documents of variable length. The architecture uses positional encoding and unitary batch sizes to process variable length documents and uses a weight-shared Fourier network layer for low-complexity token mixing. Finally, we reach the theoretical computational complexity lower bound for relation extraction using a selective pooling strategy and distance-aware attention weights with trainable polynomial distance functions. We evaluated the JNRF architecture using the 2018 N2C2 ADE benchmark to jointly extract medication-related entities and relations in variable-length EHR summaries. JNRF outperforms rolling window BERT with selective pooling by 0.42%, while being twice as fast to train. Compared to state-of-the-art BiLSTM-CRF architectures on the N2C2 ADE benchmark, results show that the proposed approach trains 22 times faster and reduces GPU memory consumption by 1.75 folds, with a reasonable performance tradeoff of 90%, without the use of external tools, hand-crafted rules or post-processing. Given the significant carbon footprint of deep learning models and the current energy crises, these methods could support efficient and cleaner information extraction in EHRs and other types of large-scale document databases.

1 Introduction

Adverse drug events (ADEs) are defined as any injury resulting from medication use and comprise the largest category of adverse events (Leape et al., 1991; Bates et al., 1995). Serious ADEs have been estimated to cost from $30 to $137 billion in ambulatory settings in the US (Johnson and Booman, 1996), and their costs have been doubling since then (Ernst and Grizzle, 2001). Due to safety concerns, between 21% to 27% of marketed drugs in the US have received black-box warnings or have been withdrawn by the Food and Drug Administration (FDA) within the first 16 years of marketing (Frank et al., 2014).

Clinical notes stored in electronic health record (EHRs) systems are a valuable source of information for pharmacovigilance (Boland and Tatonetti, 2015). However, only 1% of ADEs recorded in EHRs are reported to ADE registries, such as the FDA Adverse Event Reporting System (FAERS), while coded diagnoses have low sensitivity for ADEs (Nadkarni, 2010; Classen et al., 2011). Recognizing medication-related entities in clinical notes, extracting relations among them, and structuring this information can help identify ADEs in early stages of the drug marketing process, thus improving patient safety (Luo et al., 2017).

The state-of-the-art for biomedical named entity recognition (NER) and relation extraction (RE) is dominated by bidirectional LSTM (Hochreiter and Schmidhuber, 1997) or BERT (Devlin et al., 2018) architectures, combined with a CRF (Lafferty et al., 2001) layer and often hand-crafted rules (Xu et al., 2017; Christopoulou et al., 2020; Wei et al., 2020; Henry et al., 2020; Fang et al., 2021). Despite the high performance of end-to-end (E2E) NER+RE models, they have some important limitations imposed by the model complexity, e.g., quadratic in terms of entity types in the CRF layer or in terms of tokens in the dot-product attention mechanisms (Sutton et al., 2012; Shen et al., 2021), which hinders their effective application in the biomedical domain due to its large number of entities and large size of free text databases.

A particularity of NER and RE for pharmacovigilance is that efficient recall of entities and relations is of utmost importance, as we would like to avoid missing a serious ADE. Nevertheless, current approaches tend to automatically discard long distance (or inter-passage) relations (Yao et al., 2019; Christopoulou et al., 2020). Moreover, EHR documents varies significantly in length, containing from a few hundred tokens for simpler patient records up to several thousand tokens for more complex patients (e.g., chronic diseases) (Henry et al., 2020). Due to their computational complexity, these methods cannot process EHRs in their integrity without resorting to impractical and/or inefficient techniques such as windowing strategies (Ding et al., 2020; Pappagari et al., 2019; Yang et al., 2016).

Ongoing research is predominantly performance-driven, leading to a resurgence of resource-intensive models, neglecting the carbon footprint of deep learning models in favor of often marginal improvement in effectiveness (Wei et al., 2020; Knafou et al., 2020; Copara et al., 2020; Copara Zea et al., 2020; Fang et al., 2021; Naderi et al., 2021). As a consequence of the technical constraints induced by highly complex models, these methods are currently being associated to a significant excess on carbon emissions (Gibney, 2022). The most direct impact of training and deploying a machine learning model is the emission of greenhouse gases due to the increased hardware energy consumption (Ligozat and Luccioni, 2021). Therefore, a direct way to reduce the ecological impact of training and deploying machine learning models is to reduce the training and inference time, i.e., providing the community with low memory and computational cost models.

To tackle these limitations and issues, we propose the Joint-NER-RE-Fourier (JNRF) model with a reduced algorithmic complexity for information extraction. We combine positional encoding with unitary batch size training so that the model processes automatically variable size EHRs with consistent performance. We use a Fourier network to contextualize tokens with fair time and space complexity, allowing to process long documents with low-resource hardware and avoid rolling window strategies. Finally, we reach the theoretical computational complexity lower bound for relation extraction using a selective pooling strategy and distance-aware attention weights with trainable polynomial distance functions. The main contributions of this paper are as follows:

- We propose a general, lightweight, and efficient model to jointly detect clinical entities and multiple relations, while requiring low computational power and memory, without the use of external tools or handcrafted rules. The code is available at https://github.com/ds4dh/JNRF.
- We show that this model can be applied to variable length documents, without any architectural changes. More importantly, it has robust performance independent of the document size.
- To the best of our knowledge, this is the first effort to model ADE and medication extraction at the document level. Unlike existing models in the literature, we demonstrate that our approach is able to identify inter-passage relations without the need of window/input size tuning, post-processing or any further engineering.

2 Related work

The main methods to produce E2E information extraction systems are the so called *pipeline* (Sorokin and Gurevych, 2017; Chapman et al., 2018; Christopoulou et al., 2020) and *joint modeling* (Xu et al., 2017; Wei et al., 2020; Bekoulis et al., 2018; Nguyen and Verspoor, 2019; Luan et al., 2019; Wadden et al., 2019). The pipeline method consists of training two independent modules, one for NER and one for RE. These models naturally suffer from cascading errors, as the error signal from one module is not back-propagated to the other. Joint modeling aims to overcome this shortcoming by learning a unique model on a combination of NER and RE losses. Joint modeling tends to outperform pipeline methods, consistently achieving state-of-the-art performance (Wei et al., 2020; Fang et al., 2021; Bekoulis et al., 2018; Nguyen and Verspoor, 2019; Luan et al., 2019; Wadden et al., 2019). In addition, joint modeling techniques have some major advances as they allow to train two models at the same time, saving time and computation, and minimizing engineering efforts. In both cases, the E2E approach has been dominated by LSTM-CRF architectures (Xu et al., 2017; Christopoulou et al., 2020; Wei et al., 2020; Henry et al., 2020). However, they suffer from two main limitations: *i)* the computational complexity of the CRF layer (Jeong et al., 2009); and *ii)* the auto-regressive nature of the LSTM model, which prevents full parallel training (Xu et al., 2021).

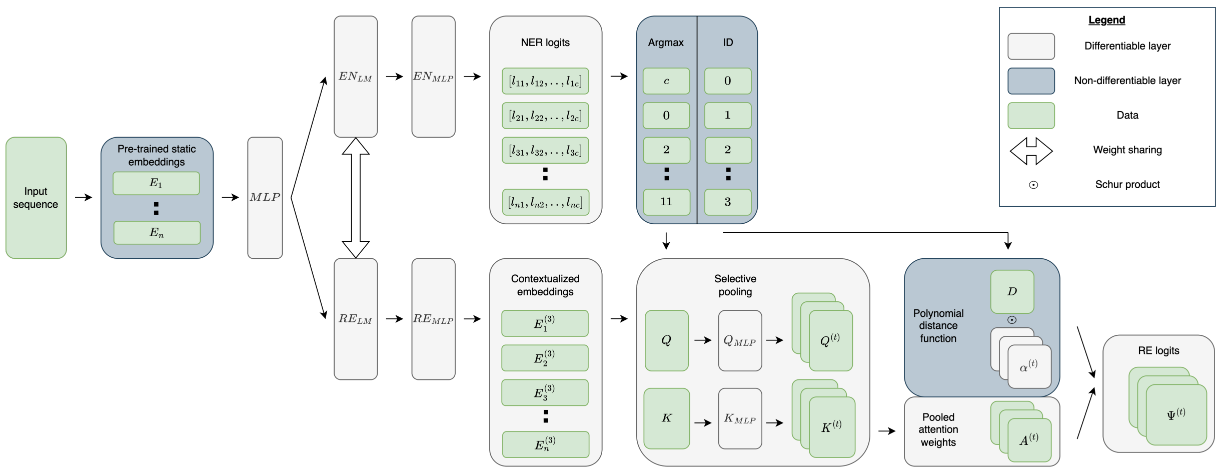


Figure 1: Computational graph for the proposed JNRF network.

2.1 Joint learning in the general domain

Bekoulis et al. (2018) proposed a joint neural model using CRFs and a multi-headed selection module allowing for multiple relation detection. The model requires the computation of scores on every pair of input tokens, which consumes $\mathcal{O}(n^2)$ time and space. To improve generalisation, their approach does not rely on external NLP tools, such as part-of-speech (POS) tagger or dependency parser. More recently, Nguyen and Verspoor (2019) proposed a joint BiLSTM-CRF architecture combined with a biaffine attention mechanism (Dozat and Manning, 2016), improving upon Bekoulis et al. (2018) in terms of time complexity. Luan et al. (2019) utilizes dynamic span graphs to learn useful information from a broader context. The graph is built by picking the most confident entity spans and linking them with confidence-weighted relation types and correlations. The model does not require pre-processing syntactic tools and significantly outperforms the previous approaches across several entity-related tasks. Lastly, DYGIE++ (Wadden et al., 2019) enumerates candidate text spans and encodes them using BERT and task-specific message updates passed over a text span graph to achieve state-of-the-art performance across entity, relation, and event extraction tasks.

2.2 Joint learning for medication-related entity and relation extraction

Most of the medication-related NER and RE studies are performed using the N2C2 ADE benchmark (Henry et al., 2020). Wei et al. (2020) proposed a system consisting of a LSTM-CRF layer for NER joint learned with a CNN-RNN layer for RE. They utilized CLAMP (Soysal et al., 2018) for the text pre-processing pipeline, including sentence boundary detection and POS labeling, and to extract a set of hand-crafted features to feed the NER module. Similarly to approaches for general corpora, Fang et al. (2021) replaced the LSTM layer by a BERT model for feature extraction, achieving 1.5 percentage point improvement in the strict F1-score metric. In their approach, a CRF layer is still used on top of a BERT model for the NER part, while a multi-head selection module (Bekoulis et al., 2018) combines the output of the BERT and CRF layers to predict relation among the detected entities.

2.3 Fourier networks

To overcome algorithmic complexity limitations in the Transformers architecture (Vaswani et al., 2017), Fourier networks (FNet) have been proposed (Lee-Thorp et al., 2021). The main innovation of FNets is that the classic Transformers attention mechanism can be mimicked using simple, non-trainable token mixing strategies. One can obtain $\mathcal{O}(n \times \log(n))$ complexity using the Cooley–Tukey Fast Fourier Transform algorithm (Cooley and Tukey, 1965) instead of the attention mechanism, which consumes $\mathcal{O}(n^2)$ with respect to the input sequence length (n). FNets achieve 92 and 97% of BERT-Base and BERT-Large (Devlin et al., 2018) accuracy on the GLUE benchmark (Wang et al., 2018), but train 70-80% faster on GPUs/TPUs. In addition to matching the accuracy of competing linear-complexity transformers (Wang et al., 2020; Jaegle et al., 2021; Wu et al., 2021; Lee-Thorp et al., 2021), the FNet is faster and memory efficient due to the unparameterized contextualization layer, i.e., it has no parameters to train for token mixing, thus requires virtually no memory usage.

3 Approach

In this section, we provide a step-by-step formal description of the proposed architecture using the forward pass representations and operations, as illustrated in Figure 1. First, we describe *i)* the vectorial token representation strategy, then *ii)* the language/contextualization layer, next *iii)* how the NER and RE task is jointly modelled, and finally *iv)* the cost functions used. Lastly, we conduct a computational complexity analysis of the proposed model.

3.1 Model formalisation

Token representation layer: We use static embeddings (BioClinicalBERT-base (Alsentzer et al., 2019) in our experiments) and freeze these parameters during training for better generalization. We also decided to use positional encoding as in Vaswani et al. (2017) so as not to fix a predefined input length.

Language model: We use FNets to perform token contextualization with fair time and space complexity. We integrate a FNet layer in our architecture as follows:

$$\begin{aligned} E^{(1)} &= \text{MLP}(E), \\ E^{(2a)} &= \text{EN}_{LM}(E^{(1)}), \\ E^{(2b)} &= \text{RE}_{LM}(E^{(1)}), \end{aligned}$$

where $E \in \mathbb{R}^{n \times d}$ is the embedding matrix, in which each row represents a token, following their order in the input sequence (i.e., the document), n the input sequence length, d the token embedding dimension, MLP is a token-wise multilayer perceptron, EN_{LM} and RE_{LM} are NER and RE FNets respectively. In fact, we fully share the weights between EN_{LM} and RE_{LM} to further reduce the number of trainable parameters. We use superscripts ($^{(1)}$, $^{(2a)}$, ...) to denote the transformed versions of the original embedding matrix.

NER and RE layers: We thus have $E^{(2)} = E^{(2a)} = E^{(2b)}$, and subsequently compute:

$$\begin{aligned} l &= \text{EN}_{MLP}(E^{(2)}), \\ E^{(3)} &= \text{RE}_{MLP}(E^{(2)}), \end{aligned}$$

where EN_{MLP} and RE_{MLP} are two independent token-wise MLPs. EN_{MLP} maps the contextualized embeddings $E^{(2)}$ to logits $l \in \mathbb{R}^{n \times c}$ for classification, where c is the number of entity classes, and RE_{MLP} maps $E^{(2)}$ to a third version of the embedding matrix $E^{(3)}$. We then compute a priori token classes

$$a_i = \text{argmax}(l_i),$$

for $i : 1 \dots n$, and apply a selective pooling strategy, i.e., we pool candidate entities for relation extraction from $E^{(3)}$ using a_i. Some relations may never exist for a particular relation extraction task. We use L to denote the set of entities that can only be linked to those of a set H. To avoid generating impossible candidate pairs, we perform two selective pooling for these two different sets: the key $K \in \mathbb{R}^{|L| \times d}$, and the query $Q \in \mathbb{R}^{|H| \times d}$. We then produce t heads

$$\begin{aligned} K^{(j)} &= \text{K}^{(j)}_{MLP}(K), \\ Q^{(j)} &= \text{Q}^{(j)}_{MLP}(Q), \end{aligned}$$

for $j : 1 \dots t$, where $\text{K}^{(j)}_{MLP}$ and $\text{Q}^{(j)}_{MLP}$ are token-wise MLPs, and t represent the number of relation types. We then compute the scores between the query and the key entities

$$A^{(j)} = Q^{(j)} K^{T(j)}.$$

As the RE module is distance agnostic, we incorporate a trainable polynomial distance function to modify the logits as a function of distance between tokens:

$$\Psi^{(j)} = A^{(j)} + \alpha_{j1} \times D^2 + \alpha_{j2} \times D + \alpha_{j3} \times I,$$

where $D_{\phi\psi}$ represents the number of tokens separating the ϕ^{th} and ψ^{th} pooled entities in the original input embedding matrix. The α's are learned through the minimization of the loss function and thus requires no predefined hand-crafted rules regarding short/long-distance relations.

Loss function: We use a cross-entropy loss for both NER and RE:

$$\mathcal{L}_{NER} = -\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{c} s(l_{i,k}) \times e_{i,k},$$

$$\mathcal{L}_{RE} = -\frac{1}{|H||L|} \sum_{h=1}^{|H|} \sum_{p=1}^{|L|} \sum_{j=1}^{t} s(\Psi^{(j)}_{h,p}) \times r_{h,p,j},$$

where $s(x_{q,z}) = \log(\exp(x_{q,z}) / \sum_b \exp(x_{q,b}))$, and e and r are the target entities and relations, respectively. Finally, we use the sum of $\mathcal{L}_{NER}$ and $\mathcal{L}_{RE}$ as the final loss function to minimize

$$\mathcal{L} = \mathcal{L}_{NER} + \mathcal{L}_{RE}.$$

3.2 Computational complexity

The complexity of the RE model depends on the number of neighbors considered for candidate pair of entities, independently of the method. If one wants to detect relations between two entities regardless of the distance, then the lower bound is $\mathcal{O}(t \times |H| \times |L|)$; or $\min(\mathcal{O}(t \times |L|)\ ,\ \mathcal{O}(t \times |H|))$ if one fixes the number of candidate neighbors. We decided not to set a maximum number of neighbors for candidate pair generation. Thus, the RE model uses $\mathcal{O}(t \times |H| \times |L|)$ through selective pooling. For a fixed RE method, the complexity of the whole model is driven by the NER component. We achieved fair complexity by using an FNet ($\mathcal{O}(n \times \log(n))$). Additionally, we used a softmax layer in place of CRF, which uses $\mathcal{O}(n \times c)$ instead of CRF's $\mathcal{O}(n \times c^2)$. This method also takes advantage of parallelization, making it a time complexity optimised method.

4 Benchmark dataset

We used the 2018 N2C2 ADE dataset [1] to evaluate our model. The data consists of 505 annotated discharge summaries from MIMIC-III (Johnson et al., 2016). The passages contains annotations for *strength*, *form*, *dosage*, *frequency*, *route*, *duration*, *reason*, and *ADE* entities, each associated with a *drug* entity. We used the official splits to train and evaluate our model, with 303 records for training and 202 for testing. Data summary statistics are presented in the Appendix A.1. *Duration* and *ADE* entities and their respective relations are not as well represented in the dataset (see Table 6). The document lengths vary widely depending on the patient's clinical history (see Table 7). There is a gap of more than 10k tokens between the smallest and largest documents (224 and 13990, respectively), which is too large to use padding efficiently. Moreover, the average document size is almost 8x larger than the typical input size of standard BERT-like implementations (4045 vs 512, respectively).

[1]Dataset available at https://portal.dbmi.hms.harvard.edu/.

5 Experiments

We trained our models in three different data representation scenarios, where we use whole documents, sentences only, and a mixed configuration where we use both documents and sentences as training instances. Performance was then evaluated at both document and sentence levels for these different training scenarios. Our models were compared to baseline models based on MLP with selective pooling and a sliding window BioClinicalBERT-base model (WBERT) (Alsentzer et al., 2019) with selective pooling, both trained and evaluated using the whole documents.

We implemented our models using PyTorch and a single Tesla V100 GPU. We used Adam (Kingma and Ba, 2014), mini-batches of size 1 and 64 for documents and sentences, respectively. Models were trained using gradient accumulation to avoid using padding tokens. The final model was selected based on the best dev F1-score obtained during training. In the following, we present the results of our experiments using micro-lenient precision, recall, and F1-score using the challenge's official evaluation tool.

5.1 Data pre-processing

We split the provided training data into train and dev sets composed of 242 and 61 documents, respectively. We tokenize documents using BioClinicalBERT-base wordpiece tokenizer from HuggingFace (Wu et al., 2016; Wolf et al., 2019). For sentence-level modeling, we first tokenize sentences using Spacy (Honnibal and Montani, 2017) and then use aforementioned wordpiece algorithm. We encode the gold entity boundaries in the BIO scheme. The embedding matrix is initialized from BioClinicalBERT-base static embeddings. No other form a data pre-processing or external feature injection has been implemented.

5.2 End-to-end effectiveness

Table 1 shows the performance of the JNRF model in multiple settings. The best performance was obtained in the document-document setting, reaching an end-to-end F1-score of 80.49%, a precision of 91.65% and a recall of 71.76%. The JNRF outperformed WBERT with selective pooling by 0.42% in F1-score (0.09% in precision and 0.06% in recall), while reducing algorithmic complexity by one order of magnitude ($\mathcal{O}(n \times (\log(n) + c))$ *vs.* $\mathcal{O}(n \times (n + c))$). We hypothesize that using

WBERT does not improve the performance due to the lack of long-range token mixing and/or an inappropriate windowing strategy. We believe that further investigation of an optimal windowing strategy could improve its performance. Moreover, we observed a significant drop in performance (37% in F1-score) when the Fnet is replaced by an MLP, demonstrating the capacity of the FNet to better attend to the correct token representations.

The JNRF model shows good performance when it is trained and evaluated with the same document representation (i.e., document-document or sentence-sentence) with similar precision in both cases and reduction in recall for the sentence-sentence setup, due to the model's limitation to detect inter-sentence relations. It is unclear though whether further data engineering could still result in equivalent performance. For the mixed training setup, the model shows stronger power to infer at the sentence level. We believe this is due to the much higher number of examples at the sentence level, which bias the model towards such representation.

Train	Language model	Test	Precision (%)	Recall (%)	F1 (%)
doc.	MLP	doc.	54.19	35.49	42.89
doc.	WBERT	doc.	90.66	71.70	80.07
doc.	FNet	doc.	**91.65**	**71.76**	**80.49**
		sent.	75.28	0.29	0.57
sent.	FNet	doc.	29.55	21.42	24.84
		sent.	89.50	65.80	75.84
mixed	FNet	doc.	66.99	32.83	44.07
		sent.	81.63	62.35	70.70

Table 1: Lenient micro-averaged E2E scores for different language models and document representations.

5.3 End-to-end efficiency

To compare the efficiency of our approach against architectures used in state-of-the-art approaches, we measured the time and memory used during training over 10 epochs (for the same training set) for a rolling window BERT (WBERT), a rolling window BERT-CRF (WBERT-CRF), and a BiLSTM-CRF. All window-based models used non-overlapping windows of size 512. We deliberately chose to use the minimum number of windows for these models to make them as fast as possible. Figure 2 shows the time and VRAM used by our model and state-of-the-art models. Results show that our model substantially improves upon the state-of-the-art in terms of time complexity. Forward and backward passes over the training dataset take an average of 30 seconds with our proposed architecture, while the average time for the above mentioned models is 54, 168 and 685 seconds, respectively. This increases the learning speed by a factor of 2, 6 and 22, respectively (Figure 2a). In addition, we measured an average VRAM usage of 8 GB for the JNRF architecture while the average memory usage for the above mentioned models is 4, 5 and 14 GBs, respectively. This represents a 43% GPU memory saving compared to BiLSTM-CRF (Figure 2b). WBERT and WBERT-CRF uses around 2x less memory due to the windowing strategy. This increase in efficiency is due to the fact that, differently from the quadratic complexity in terms of the number of entities c, which is generally large in the biomedical field, our model complexity has a linear dependency in terms of the number of entities, and a log-linear dependency in terms of the number of tokens (overall $\mathcal{O}(n \times (\log(n) + c))$).

5.4 Time inefficiency of windowing strategies

To demonstrate that windowing strategies are time inefficient, we measured the average forward-backward time of a rolling window JNRF (WJNRF) and its average VRAM usage (Figure 2). JNRF is 20% faster than WJNRF but WJNRF uses 26% less memory (Figure 2). While windowing strategies save VRAM, they are an inefficient solution in terms of computation time. The average document size is 4045 (see Table 7) corresponding to an average of 8 forward passes per document using standard BERT-like implementations (512 tokens maximum input size) or 28 for the longest document. So that all tokens attend to each other, we would need overlapping windows. The worst case scenario is to drag the window token-by-token, leading to 3534 ($n - WindowSize + 1$) windows on average per document.

5.5 Performance across entities, relations and document sizes

Table 2 shows the performance of our model per entity and relation types. Our model suffers from poor performance in extracting *Reason* and *ADE* entities, with an F1-score of 50.26% and 16.40%, respectively. This lower performance is also seen in other competing solutions (Henry et al., 2020). In turn, both the detection of their respective rela-

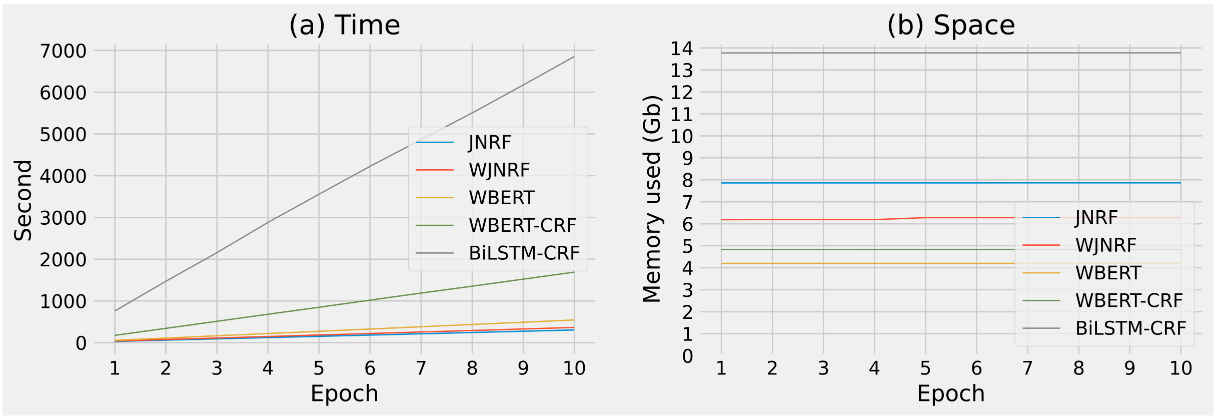


Figure 2: (a) Cumulative training time of JNRF *vs.* WJNRF *vs.* WBERT *vs.* WBERT-CRF *vs.* BiLSTM-CRF. (b) GPU memory usage of JNRF *vs.* WJNRF *vs.* WBERT *vs.* WBERT-CRF *vs.* BiLSTM-CRF. For fair comparison, all systems use the selective pooling RE module.

tions are also negatively impacted, with a final E2E F1-score of only 29.92% and 7.21%, respectively. We believe this lower performance is a result of the confusion between these entities (as they are semantically similar) and of the small number of instances in the training set. Nevertheless, further investigation is needed to better understand the issue.

Entity	Precision (%)	Recall (%)	F1 (%)
Drug	93.32	86.99	90.05
Strength	96.80	95.08	95.93
Form	96.57	92.38	94.43
Dosage	94.26	87.62	90.82
Frequency	97.63	92.37	94.93
Route	92.63	93.03	92.83
Duration	84.98	61.38	71.27
Reason	65.96	40.59	50.26
ADE	36.67	10.56	16.40
Overall	92.95	84.76	88.67
Entity + Relation	**Precision (%)**	**Recall (%)**	**F1 (%)**
Strength-Drug	95.60	88.60	91.97
Form-Drug	95.63	87.01	91.12
Dosage-Drug	94.13	79.07	85.94
Frequency-Drug	94.83	83.19	88.63
Route-Drug	90.50	83.25	86.72
Duration-Drug	76.09	41.08	53.35
Reason-Drug	54.72	20.59	29.92
ADE-Drug	30.30	4.09	7.21
Overall	90.97	72.08	80.43

Table 2: NER and E2E (NER+RE) performance of our JNRF model.

Table 3 shows the performance as a function of the number of input tokens (document length). We followed the Freedman-Diaconis method (Freedman and Diaconis, 1981) to group documents into clusters of different lengths. These results highlights the ability of our architecture to perform consistently across clinical notes of varying sizes. Without any data pre-processing (e.g., sliding window or sentence tokenization), the model can elegantly generalise to document of different sizes.

Doc. length	Doc. count	Precision (%)	Recall (%)	F1 (%)
[0, 754]	5	91.67	59.46	72.13
[754, 1508]	8	97.25	67.22	79.49
[1508, 2262]	18	89.92	66.55	76.49
[2262, 3016]	28	91.65	74.69	82.30
[3016, 3770]	43	91.72	73.46	81.58
[3770, 4524]	30	90.35	71.64	79.91
[4524, 5278]	32	90.82	72.60	80.69
[5278, 6032]	18	89.58	72.16	79.93
[6032, 6786]	10	93.88	70.99	80.85
[6786, 7540]	4	89.17	70.01	78.44
[7540, 8294]	3	88.89	67.06	76.45
[8294, 9048]	1	92.08	75.61	83.04
[9802, 10556]	1	88.73	66.55	76.06
[12064, 12818]	1	92.54	80.84	86.30

Table 3: Performance of our JNRF model across different document sizes.

5.6 Performance on long range relations

Figure 3 shows the distribution of relation types according to their sentence distance. We define the sentence distance between two related entities $E1$

and $E2$ as the number of sentences separating $E1$ from $E2$. A negative distance implies that the *drug* entity is mentioned before the related entity. Results show that although most related entities are in the same sentence, there are a non-negligible number of relations with a sentence distance different from zero. As we can see from Table 4, the JNRF model is able to automatically detect distant relations. It has superior performance detecting intra-sentence relations, i.e., better F1-score for sentence distance 0, with a yet robust performance for inter-sentence relations with negative sentence distances (between 65% and 68% F1-score). The performance decreases substantially for inter-sentence relations with positive sentence distances. This is due to the fact that *Reason* and *ADE* entities and relations are actually harder to detect (see Table 2), and they represent the vast majority of relations with a positive sentence distance, as shown in Figure 3. It is important to note that using a fixed-input size models would only detect intra-sentence relations or inter-sentence through significant engineering, which may not necessarily generalise to other corpora and domains.

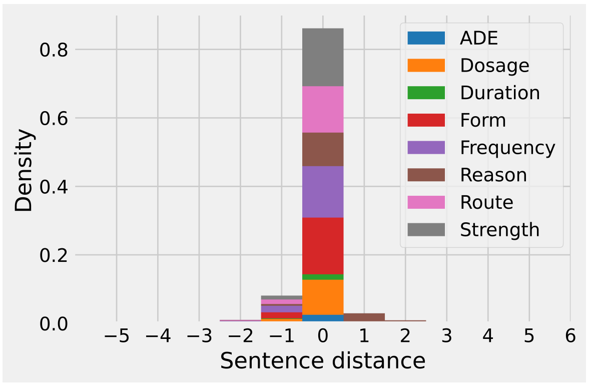


Figure 3: Probability density estimation of relation types as a function of the number of sentences separating two related entities (Sentence distance).

	Sentence distance				
	-2	**-1**	**0**	**1**	**2**
Precision (%)	75.14	83.06	92.69	22.99	0.36
Recall (%)	56.90	57.88	76.08	5.82	0.49
F1-score (%)	64.76	68.22	83.57	9.29	0.41

Table 4: Performance of our JNRF model as a function of sentence distance.

6 Comparison with SOTA in the N2C2 ADE challenge

In this section, for a reference we show our results against state-of-the-art E2E NER+RE models described in the N2C2 ADE challenge (Henry et al., 2020). Nevertheless, due to their different modelling strategy (e.g., multiple models, external tools, post-processing techniques and hand-crafted rules specifically designed for this dataset), they are not directly comparable.

UTH (Wei et al., 2020) used a joint learning model consisting of a LSTM-CRF layer for NER and a CNN-RNN layer for RE. CLAMP (Soysal et al., 2018) was employed for text pre-processing, including sentence boundary detection and POS labeling, and to create a set of hand-crafted features that fed the CRF layer. Entities without a relation were associated to the closest drug in the post-processing step.

NaCT (Christopoulou et al., 2020) used a majority voting ensemble of feature-based CRF, including ADE dictionary, and stacked BiLSTM-CRF for NER. For RE, they used an ensemble of LSTM for intra-sentence relations and a transformer network for inter-sentence relations.

BCH (Miller et al., 2019) used SVM to detect entities, and pair these detected entities for a second SVM relation classifier. They used cTAKES (Savova et al., 2010) to pre-process data and ClearTK (Bethard et al., 2014) API to extract features.

RA (Henry et al., 2020) used dictionary-based features, CRFs and logistic regression for NER. For RE, they used a tree-based boosting classifier (Chen and Guestrin, 2016).

Table 5 shows the performance of our best model as well as the results of the previously described systems. As we can see, the performance of our E2E model (80.49% F1-score) achieves 90% of the F1-score of the best performing system (99% precision and 84% recall), while significantly reducing algorithmic complexity. Moreover, it compares favorably to strong baseline methods (Chen and Guestrin, 2016) (80.49% *vs.* 80.37%), again with an order of magnitude in complexity reduction.

Name	NER complexity	Precision (%)	Recall (%)	F1 (%)
UTH	nc^2	**92.92**	**85.49**	**89.05**
NaCT	nc^2	92.64	83.18	87.66
BCH	n^3	89.63	76.40	82.49
JNRF	$n(\log(n) + c)$	91.65[a]	71.76[b]	80.49[c]
RA	nc^2	86.89	74.75	80.37

Table 5: E2E scores of the top performing systems submitted in the N2C2 ADE track, along with our JNRF model. Standard deviations: a=0.47, b=0.53, c=0.33.

7 Conclusion

In this paper, we proposed an end-to-end, generalizable, lightweight, and efficient model to jointly detect entities and multiple relations at the intra- and inter-passage levels. We combined a Fourier network with a pooled attention layer to significantly reduce time and space complexity, thus providing the community with a low carbon footprint solution for end-to-end relation extraction. We demonstrated that our model outperformed the sliding window BERT with selective pooling by 0.42% in F1-score, while being 2 times faster to train. Furthermore, we showed that our model trains 22 times faster and consumes 1.75 times less GPU memory than state-of-the-art BiLSTM-CRF architectures, with a reasonable performance tradeoff of 90% on the N2C2 ADE benchmark, without using external tools or hand-crafted rules. Furthermore, we showed that this approach achieves consistent performance regardless of the length of the input sequence, eliminating the need for sliding window techniques and easing the overall data processing pipeline and engineering effort.

References

Emily Alsentzer, John R Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew McDermott. 2019. Publicly available clinical bert embeddings. *arXiv preprint arXiv:1904.03323*.

David W Bates, David J Cullen, Nan Laird, Laura A Petersen, Stephen D Small, Deborah Servi, Glenn Laffel, Bobbie J Sweitzer, Brian F Shea, Robert Hallisey, et al. 1995. Incidence of adverse drug events and potential adverse drug events: implications for prevention. *Jama*, 274(1):29–34.

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018. Joint entity recognition and relation extraction as a multi-head selection problem. *Expert Systems with Applications*, 114:34–45.

Steven Bethard, Philip Ogren, and Lee Becker. 2014. ClearTK 2.0: Design patterns for machine learning in UIMA. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3289–3293, Reykjavik, Iceland. European Language Resources Association (ELRA).

Mary Regina Boland and Nicholas P Tatonetti. 2015. Are all vaccines created equal? using electronic health records to discover vaccines associated with clinician-coded adverse events. *AMIA Summits on Translational Science Proceedings*, 2015:196.

Alec B Chapman, Kelly S Peterson, Patrick R Alba, Scott L DuVall, and Olga V Patterson. 2018. Hybrid system for adverse drug event detection. In *International Workshop on Medication and Adverse Drug Event Detection*, pages 16–24. PMLR.

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.

Fenia Christopoulou, Thy Thy Tran, Sunil Kumar Sahu, Makoto Miwa, and Sophia Ananiadou. 2020. Adverse drug events and medication relation extraction in electronic health records with ensemble deep learning methods. *Journal of the American Medical Informatics Association*, 27(1):39–46.

David C Classen, Roger Resar, Frances Griffin, Frank Federico, Terri Frankel, Nancy Kimmel, John C Whittington, Allan Frankel, Andrew Seger, and Brent C James. 2011. 'global trigger tool'shows that adverse events in hospitals may be ten times greater than previously measured. *Health affairs*, 30(4):581–589.

James W. Cooley and John W. Tukey. 1965. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301.

Jenny Copara, Julien Knafou, Nona Naderi, Claudia Moro, Patrick Ruch, and Douglas Teodoro. 2020. Contextualized french language models for biomedical named entity recognition. In *6e conférence conjointe Journées d'Études sur la Parole (JEP, 33e édition), Traitement Automatique des Langues Naturelles (TALN, 27e édition), Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RÉCITAL, 22e édition). Atelier DÉfi Fouille de Textes*, pages 36–48. ATALA; AFCP.

Jenny Linet Copara Zea, Nona Naderi, Julien David Marc Knafou, Patrick Ruch, and Douglas Teodoro. 2020. Named entity recognition in chemical patents using ensemble of contextual language models. In *Proceedings of CLEF (Conference and Labs of the Evaluation Forum) 2020 Working Notes*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*.

Ming Ding, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. Cogltx: Applying bert to long texts. In *Advances in Neural Information Processing Systems*, volume 33, pages 12792–12804. Curran Associates, Inc.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Frank R Ernst and Amy J Grizzle. 2001. Drug-related morbidity and mortality: updating the cost-of-illness model. *Journal of the American Pharmaceutical Association (1996)*, 41(2):192–199.

Xintao Fang, Yuting Song, and Akira Maeda. 2021. Joint extraction of clinical entities and relations using multi-head selection method. In *2021 International Conference on Asian Language Processing (IALP)*, pages 99–104. IEEE.

Cassie Frank, David U Himmelstein, Steffie Woolhandler, David H Bor, Sidney M Wolfe, Orlaith Heymann, Leah Zallman, and Karen E Lasser. 2014. Era of faster fda drug approval has also seen increased black-box warnings and market withdrawals. *Health affairs*, 33(8):1453–1459.

David Freedman and Persi Diaconis. 1981. On the histogram as a density estimator: L 2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 57(4):453–476.

Elizabeth Gibney. 2022. How to shrink ai's ballooning carbon footprint. *Nature*, 607(7920):648–648.

Sam Henry, Kevin Buchan, Michele Filannino, Amber Stubbs, and Ozlem Uzuner. 2020. 2018 n2c2 shared task on adverse drug events and medication extraction in electronic health records. *Journal of the American Medical Informatics Association*, 27(1):3–12.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. 2021. Perceiver: General perception with iterative attention. *arXiv preprint arXiv:2103.03206*.

Minwoo Jeong, Chin-Yew Lin, and Gary Geunbae Lee. 2009. Efficient inference of CRFs for large-scale natural language data. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 281–284, Suntec, Singapore. Association for Computational Linguistics.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9.

Jeffery Johnson and Lyle Booman. 1996. Drug-related morbidity and mortality. *Journal of Managed Care Pharmacy*, 2(1):39–47.

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

Julien David Marc Knafou, Nona Naderi, Jenny Linet Copara Zea, Douglas Teodoro, and Patrick Ruch. 2020. Bitem at wnut 2020 shared task-1: Named entity recognition over wet lab protocols using an ensemble of contextual language models. In *Proceedings of the 2020 EMNLP Workshop W-NUT: The Sixth Workshop on Noisy User-generated Text*, pages 305–313. Association for Computational Linguistics.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Lucian L Leape, Troyen A Brennan, Nan Laird, Ann G Lawthers, A Russell Localio, Benjamin A Barnes, Liesi Hebert, Joseph P Newhouse, Paul C Weiler, and Howard Hiatt. 1991. The nature of adverse events in hospitalized patients: results of the harvard medical practice study ii. *New England journal of medicine*, 324(6):377–384.

James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. 2021. Fnet: Mixing tokens with fourier transforms. *arXiv preprint arXiv:2105.03824*.

Anne-Laure Ligozat and Sasha Luccioni. 2021. *A Practical Guide to Quantifying Carbon Emissions for Machine Learning Researchers and Practitioners*. Ph.D. thesis, MILA; LISN.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. 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 3036–3046, Minneapolis, Minnesota. Association for Computational Linguistics.

Yuan Luo, William K Thompson, Timothy M Herr, Zexian Zeng, Mark A Berendsen, Siddhartha R Jonnalagadda, Matthew B Carson, and Justin Starren. 2017. Natural language processing for ehr-based pharmacovigilance: a structured review. *Drug safety*, 40(11):1075–1089.

Timothy Miller, Alon Geva, and Dmitriy Dligach. 2019. Extracting adverse drug event information with minimal engineering. In *Proceedings of the 2nd Clinical*

Natural Language Processing Workshop, pages 22–27, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Nona Naderi, Julien Knafou, Jenny Copara, Patrick Ruch, and Douglas Teodoro. 2021. Ensemble of deep masked language models for effective named entity recognition in health and life science corpora. *Frontiers in research metrics and analytics*, 6.

Prakash M Nadkarni. 2010. Drug safety surveillance using de-identified emr and claims data: issues and challenges. *Journal of the American Medical Informatics Association*, 17(6):671–674.

Dat Quoc Nguyen and Karin Verspoor. 2019. End-to-end neural relation extraction using deep biaffine attention. In *European Conference on Information Retrieval*, pages 729–738. Springer.

Raghavendra Pappagari, Piotr Zelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. 2019. Hierarchical transformers for long document classification. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 838–844. IEEE.

Guergana K Savova, James J Masanz, Philip V Ogren, Jiaping Zheng, Sunghwan Sohn, Karin C Kipper-Schuler, and Christopher G Chute. 2010. Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association*, 17(5):507–513.

Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. 2021. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3531–3539.

Daniil Sorokin and Iryna Gurevych. 2017. Context-aware representations for knowledge base relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1784–1789, Copenhagen, Denmark. Association for Computational Linguistics.

Ergin Soysal, Jingqi Wang, Min Jiang, Yonghui Wu, Serguei Pakhomov, Hongfang Liu, and Hua Xu. 2018. Clamp–a toolkit for efficiently building customized clinical natural language processing pipelines. *Journal of the American Medical Informatics Association*, 25(3):331–336.

Charles Sutton, Andrew McCallum, et al. 2012. An introduction to conditional random fields. *Foundations and Trends® in Machine Learning*, 4(4):267–373.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5784–5789, Hong Kong, China. Association for Computational Linguistics.

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*.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Qiang Wei, Zongcheng Ji, Zhiheng Li, Jingcheng Du, Jingqi Wang, Jun Xu, Yang Xiang, Firat Tiryaki, Stephen Wu, Yaoyun Zhang, et al. 2020. A study of deep learning approaches for medication and adverse drug event extraction from clinical text. *Journal of the American Medical Informatics Association*, 27(1):13–21.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Chuhan Wu, Fangzhao Wu, Tao Qi, Yongfeng Huang, and Xing Xie. 2021. Fastformer: Additive attention can be all you need. *arXiv preprint arXiv:2108.09084*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv:1609.08144*.

Hongfei Xu, Qiuhui Liu, Josef van Genabith, Deyi Xiong, and Meng Zhang. 2021. Multi-head highly parallelized lstm decoder for neural machine translation. 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 273–282.

Jun Xu, Hee-Jin Lee, Zongcheng Ji, Jingqi Wang, Qiang Wei, and Hua Xu. 2017. Uth_ccb system for adverse drug reaction extraction from drug labels at tac-adr 2017. In *TAC*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. Docred: A large-scale document-level relation extraction dataset. *arXiv preprint arXiv:1906.06127*.

A Appendix

A.1 N2C2 dataset summary statistics

Entity type	**Full (%)**	**Training**	**Test**
Drug	26.8k (32)	16.2k	10.6k
Strength	10.9k (13)	6.7k	4.2k
Form	11.0k (13)	6.7k	4.4k
Dosage	6.9k (8)	4.2k	2.7k
Frequency	10.3k (12)	6.3k	4.0k
Route	9.0k (11)	5.5k	3.5k
Duration	1.0k (1)	0.6k	0.4k
Reason	6.4k (8)	3.9k	2.5k
ADE	16k (2)	1.0k	0.6k
Total	83.8k (100)	51.0k	32.9k
Relation type	**Full (%)**	**Training**	**Test**
Strength-Drug	10.9k (18)	6.7k	4.2k
Form-Drug	11.0k (19)	6.7k	4.4k
Dosage-Drug	6.9k (11)	4.2k	2.7k
Frequency-Drug	10.3k (17)	6.3k	4.0k
Route-Drug	9.1k (15)	5.5k	3.5k
Duration-Drug	1.1k (2)	0.6k	0.4k
Reason-Drug	8.6 (15)	5.2k	3.4k
ADE-Drug	1.8 (3)	1.1k	0.7k
Total	59.8 (100)	36.4k	23.5k

Table 6: Entity and relation distributions.

	Train set	**Validation set**	**Test set**
Count	242	61	202
Mean	4045	3829	3933
Std	1972	1870	1790
Min	224	237	252
Max	13990	7845	12518

Table 7: Statistics of document length in terms of tokens.

Genre Transfer in NMT:
Creating Synthetic Spoken Parallel Sentences using Written Parallel Data

Nalin Kumar and **Ondřej Bojar**
Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics
Charles University, Prague, Czech Republic
knalin55@gmail.com, bojar@ufal.mff.cuni.cz

Abstract

Text style transfer (TST) aims to control attributes in a given text without changing the content. The matter gets complicated when the boundary separating two styles gets blurred. We can notice similar difficulties in the case of parallel datasets in spoken and written genres. Genuine spoken features like filler words and repetitions in the existing spoken genre parallel datasets are often cleaned during transcription and translation, making the texts closer to written datasets. This poses several problems for spoken genre-specific tasks like simultaneous speech translation. This paper seeks to address the challenge of improving spoken language translations. We start by creating a genre classifier for individual sentences and then try two approaches for data augmentation using written examples: (1) a novel method that involves assembling and disassembling spoken and written neural machine translation (NMT) models, and (2) a rule-based method to inject spoken features. Though the observed results for (1) are not promising, we get some interesting insights into the solution. The model proposed in (1) fine-tuned on the synthesized data from (2) produces naturally looking spoken translations for written→spoken genre transfer in En-Hi translation systems. We use this system to produce a second-stage En-Hi synthetic corpus, which however lacks appropriate alignments of explicit spoken features across the languages. For the final evaluation, we fine-tune Hi-En spoken translation systems on the synthesized parallel corpora. We observe that the parallel corpus synthesized using our rule-based method produces the best results.

1 Introduction

Style transfer has been one of the well-studied tasks in the field of Artificial Intelligence (AI). Most of the earlier works using deep learning were in the domain of Computer Vision (CV; Gatys et al., 2016; Zhu et al., 2017). The task of text style transfer (TST) saw a surge in research interests after the inception of attention-based sequence-to-sequence text generation models. The essence of any of the tasks under TST is bringing changes in certain stylistic attributes while preserving the content. One such task is creating synthetic spoken parallel data using a written one (Figure 1). The task is novel, and minimal work is publicly available where both spoken and written genres are addressed distinctly.

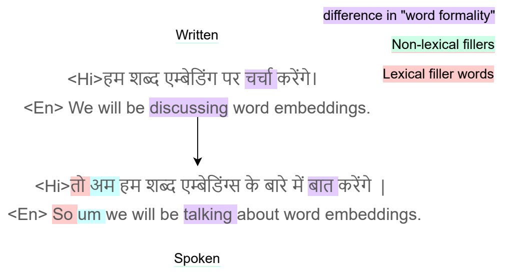

difference in "word formality"
Non-lexical fillers
Lexical filler words
Written
<Hi>हम शब्द एम्बेडिंग पर चर्चा करेंगे।
<En> We will be discussing word embeddings.
<Hi>तो अम हम शब्द एम्बेडिंग्स के बारे में बात करेंगे ।
<En> So um we will be talking about word embeddings.
Spoken

Figure 1: Text style transfer from written to spoken parallel sentences.

Because of its easy maintenance and availability, written data have been extensively experimented with. This works for most tasks; however, with the increasing popularity of processing speech, such as simultaneous translation of spoken language, the need for speech-specific data grows. However, transcribing large volumes of audio datasets is a tedious and costly process. In addition, creating a parallel dataset for such tasks requires the further step of translation. Applying methods of TST can offer a solution to this problem by utilizing the available large amount of written parallel text.

The utilization of written parallel data for the creation of spoken ones is not straightforward. A major issue with this task is having a clear distinction between spoken and written genres. Spoken genre spans a broad spectrum with spontaneous conversations or speeches at one extreme and prepared speeches at the other one. The lack of spontaneity in the latter case can make sentences indistinguishable from the written genre. The existing parallel datasets rarely contain genuine examples of spontaneous speech, as most of the filler words and pauses are cleaned while transcribing audio samples.

Most works in the field do not address the challenge of transfer between two similar linguistic styles. Since most of them are based on data-driven approaches, they are prone to failing in this setting due to insufficiently distinct features of the two genres. The common alternative, simple rule-based approaches model only content-independent features. We thus see a need for a combination of both approaches.

The main objective of this paper is to improve spoken translation systems. We address the lack of parallel spontaneous speech corpora using TST from written genre to spoken genre in the context of NMT. In this work, we try different data augmentation methods to construct synthetic spoken-style parallel dataset from existing written genre parallel datasets. We inject disfluencies in both languages on the phrase level. Even though not all disfluencies are consistent across languages in real life, we try to preserve them in the constructed parallel data, at least on the phrase level.

We propose a data-driven approach involving NMT models from both genres. We combine the encoders and decoders extracted from translation systems trained on each of the genres separately. We also propose a rule-based method for constructing spoken-style parallel data by injecting spoken features at the phrase-level to existing written parallel datasets. We check the applicability of the created synthetic parallel datasets for Hindi-English spoken language translation systems. To summarize, the main contributions of this paper are:

- We propose and implement a genre classifier for individual sentences.
- We propose a seq-to-seq model for translating from written genre to spoken genre.
- We provide a rule-based method for synthesizing spoken-style data from written genre examples.
- We evaluate the effectiveness of our genre transfer methods on a spoken language translation system for En-Hi language pair.

We discuss work related to our approach in Section 2. We provide an overview of the used datasets in Section 3. We describe our proposed genre classifier in Section 4 and continue with our data augmentation methods in Sections 5 and 6. We provide analysis of the results in Section 7. We discuss some alternatives and future directions in Section 8 and finally conclude our paper in Section 9.

We publish our source code, and pre-trained models on GitHub. [1]

2 Related Works

Our work touches upon three topics: Neural Machine Translation (Section 2.1), Text Style Transfer (Section 2.2), and Evaluation of Stylised Texts (Section 2.3).

2.1 Neural Machine Translation (NMT)

Since the advent of encoder-decoder-based methods (Cho et al., 2014), NMT has seen an uninterrupted flow of research interests. When given a large amount of training data, it has performed significantly better than the traditional methods. There has been considerable focus on techniques such as transfer learning (Zoph et al., 2016; Lakew et al., 2018) and data augmentation (Sennrich et al., 2016; Nguyen et al., 2020; Shen et al., 2020) to tackle the problem of low-resource settings. There has also been a limited amount of work on studying stylistic features in NMT outputs. For instance, Niu et al. (2017) use the lexical formality model to control the formality level of the NMT outputs. Wu et al. (2021) propose a bidirectional knowledge transfer framework to produce stylized translations.

2.2 Text Style Transfer (TST)

TST aims to control the stylistic attributes of a given text without changing the content. Stylistic attributes can range from politeness, formality, etc., to literary writing style. Some of the earlier works include Yan (2016); Ghazvininejad et al. (2016) for style transfer in poetry, Jhamtani et al. (2017) for Shakespearizing modern English, dos Santos et al. (2018) for controlling offensive language, and many more. However, due to the lack of parallel datasets, most of the solutions in TST revolve around unsupervised methods. Replacing style-specific words is one of the trivial and earlier solutions. However, the complexity of a natural language text can make this approach visibly suboptimal. One of the popular techniques is to disentangle the content and style dimensions in the latent

[1] https://github.com/knalin55/Genre-Transfer-in-NMT

	lex_fil		nlex_fil		fp		rep		ph_abbr		mean_len	
	en	hi	en	hi	en	hi	en	hi	en	hi	en	hi
Online Lec	38	37	2	0	21	21	11	4	8	-	23	25
OpenSub	1	1	0	0	18	18	0	0	22	-	6	7
Wikipedia	0	0	0	0	1	1	0	0	1	-	18	21
VOICE 2.0	26	-	21	-	25	-	25	-	12	-	15	-

Table 1: Data sources and some of their spoken features (out of 50 randomly examples).

space representing the input text (John et al., 2019; Yamshchikov et al., 2019). Although most recent works focus on this approach, they have a rather limited control over the model. Another interesting approach, which has gained popularity recently, is supervised training on a pseudo-parallel data. Prabhumoye et al. (2018) use back-translation for semantic preservation and adversarial training to generate texts in a specific style.

2.3 Evaluation of Stylised Texts

Evaluation for TST is challenging due to the subjectivity of styles. According to Mir et al. (2019), there are broadly three aspects of evaluation: style accuracy, content preservation, and natural and fluent output. In our case, along with these three aspects, we also need to ensure the translational equivalence of the sentences. Stylistic features are often independent of a set of particular lexicons. Thus, building a rule-based classifier is not so straightforward. A deep learning-based data-driven classifier can solve the issue but it faces data scarcity. Moreover, capturing content preservation using automatic evaluation metrics is even more challenging due to their reliance on similarity of the candidate translation and the reference. With style change, this similarity can be failing. For Wu et al. (2021), the objective is similar to ours. They trained a BERT-based classifier for the classification of formality. They use a language model for checking the fluency of the translated output, BLEU (Papineni et al., 2002) for the evaluation of translated outputs, and human evaluation for checking overall quality.

There are undoubtedly multiple parameters for judging the quality of stylized texts. Thus, we need to have multiple metrics covering all aspects. In our work, use BLEU to check translation quality, a genre classifier for style quality, and manual evaluation to check fluency and overall quality.

3 Data

We use the Samanantar corpus for our experiments. We split the corpus into written and spoken parts using the data sources. We consider sources from lectures (*Coursera*, *NPTEL*, *KhanAcademy*, and *Kurzgesagt*), and *OpenSubtitles* belonging to the spoken genre, and *Wikipedia* to the written genre. There are abundant examples where sentences have lexical filler words; however, very few contain non-lexical fillers or repetitions.

We have a total of $171,416$ parallel sentences for the spoken genre and $216,183$ for the written one. We separate $5,000$ sentences as a test set, $10,000$ as validation set, and the remaining sentences as a training set. We ensure no training sentence appears in the test or validation data.

Table 1 provides some statistics for spoken features of randomly sampled 50 sentences from each of our data sources. We calculate mean length of sentences over the whole data source.

3.1 Online Lectures

Online learning platforms are a great source of spoken genre data. Though they cannot be categorized as entirely spontaneous and are certainly well-prepared, they can contain a substantial amount of filler words. We cluster the sources $KhanAcademy$, $Coursera$, $Nptel$, and $Kurzgesagt$ together as online lectures. Table 1 shows they have longer sentences, and a considerable number of them have lexical filler words (lex_fil). However, all these filler words are in the form of sentence connectors (like, so, OK, well, etc.), and almost none of the examples had non-lexical fillers (nlex_fil) like, ermm, umm, uh, etc. 42% of the examples were in first-person (fp), 22% of the En examples had repetitions (8% of Hi examples; we suspect repetitions were removed while generating translations) and only 16% contained phonological abbreviations (ph_abbr). In summary, 43 out of 50 randomly sampled En sentences contained some form of listed spoken features. Though the spoken style quality and spontaneity of the spoken sources are substandard, these examples are the best we could have for the Hi language in the given genre.

Feature	Input text	en_{voice}	en_{base}
None	She later expressed regret for having cited an inaccurate study	0	0
Sentence conn	*So* she later regretted for having cited an inaccurate study	1	1
First person	*I* later regretted for having cited an inaccurate study	1	1
Filler word	She later regretted *er erm* for having cited an inaccurate study	1	0
Filler word	She later regretted *umm* for having cited an inaccurate study	0	0
Repetition	She later regretted *for for* having cited an inaccurate study	0	0

Figure 2: En genre classifiers' predictions compared against different spoken features. The first example is of written genre (0, as predicted by both models en_{base} and en_{voice}), while the rest are from spoken genre (1, which is not always predicted).

3.2 OpenSubtitles

OpenSubtitles is a collection of multilingual parallel corpora compiled from an extensive database of movies and TV subtitles. Since the dialogues and conversations are well rehearsed and prepared, they seem to have fewer fillers and repetitions (see Table 1). They also have shorter sentences. The sentences are understandably of the spoken genre; however, the lack of fillers and other spoken features might hurt the classifier and our MT model.

3.3 Wikipedia

Wikipedia is one of the most experimented written genre datasets available in the field. As expected, none of the randomly sampled examples contained any filler words or repetition.

3.4 VOICE 2.0

We use an additional source of En spoken monolingual corpus, VOICE 2.0 (Vienna-Oxford International Corpus of English), to train our En genre classifier. VOICE 2.0 is a collection of English spoken data. We use it as our additional data source for training our En genre classifier. Table 1 shows the dataset has a considerable amount of repetitions and filler words. The fillers contained a mix of both non-lexical and lexical words. 46 of the 50 randomly sampled examples had at least one of the spoken features mentioned in table. We take 62348 En sentences from the dataset for our classification experiment.

4 Genre Classifier

Classifying genre plays a vital role in the evaluation of genre transfer tasks. Majority of the papers in TST use deep-learning-based classifiers to classify the specific style. Along the lines of existing approaches, we train a BERT-based genre classifier. We train two models: with and without VOICE 2.0 dataset for En classifier. Since we have parallel sentences, we can analyze the results for En and draw also some conclusions for Hi language.

Model	Test f1 (%)
hi_{base}	96.07
en_{base}	97.84
en_{voice}	98.12

Table 2: Performance of genre classifiers

We use DistilBERT (Sanh et al., 2019) as our pretrained model for En language. It is a fast, cheap and light Transformer model trained by distilling BERT base. It has 40% fewer parameters and is 60% faster than bert-base-uncased. We train the model on En sentences from Samanatar corpus (en_{base}) and Samanantar + VOICE 2.0 corpus (en_{voice}). We use HuggingFace trainer for our experiments. We use the same tokenizer as DistilBert and set the sequence max length to 256. We train the models for 2 epochs with a batch size of 8 while keeping the best checkpoints at each 500 steps. We do not train the model further as its performance stopped improving after 2 epochs. We use 10000 sentences as test (comprised of examples from Samanantar and VOICE 2.0), 20000 sentences as validation, and the remaining sentences in the dataset as training data.

We use IndicBERT (Kakwani et al., 2020) as our pretrained model for Hi genre classification. IndicBERT is a multilingual AlBERT (Lan et al., 2019) trained on 12 Indic languages. The model has SOTA performance when compared to other multilingual models. We train the model on Hi sentences from *OpenSubtitles*, Online Lectures, and *Wikipedia* in Samanantar corpus (hi_{base}). We keep the hyperparameters similar to the En classifiers.

4.1 Results

We evaluate our En models on the Samanantar + VOICE 2.0 dataset test set. Since the training data has examples from diverse domains, there is less possibility for any bias towards a specific domain.

The En classifiers, en_{base} and en_{voice}, give F1

Algorithm 1: Algorithm to create a spoken sentence pair using a written sentence pair

```
Input $e_{written}, f_{written}$
Procedure gen_spoken($e_{written}, f_{written}$):
    $e_{spoken}, f_{spoken} \leftarrow list(), list()$
    $alignments = get\_word\_alignments(e_{written}, f_{written})$
    $phrase\_align = get\_phrase(alignments)$
    $phrase\_align.insert((init\_filler_e, init\_filler_f), index = 0)$ with some probability $p_i$
    for $phrase \in phrase\_align$ do
        $phrase \leftarrow add\_spoken\_features(phrase)$ with some probability P
    $e_{spoken}.add(phrase_e \forall phrase_e \in phrase\_align)$
    $f_{spoken}.add(phrase_f \forall phrase_f \in phrase\_align)$
    return $e_{spoken}, f_{spoken}$
Function get_phrase($word\_alignments$):
    while $len(word\_alignments)$ does not decrease do
        for $align_i, align_{i+1} \in word\_alignments$ do
            if step == 1 then
                concat($align_i, align_{i+1}$) if $set(tgt(align_i) \bigcup tgt(align_{i+1}))$ contains consecutive indices
            else
                concat($align_i, align_{i+1}$) if $src(align_i), src(align_{i+1})$ overlap
    return $word\_alignments$
Function add_spoken_features(phrase, P):
    $fillers_e, fillers_f$ = set of filler words in lang $e$, lang $f$
    with prob $\mathbf{P}_{fill}$, $phrase_e, phrase_f \leftarrow phrase_e.append(fillers_e[i]), phrase_f.append(fillers_f[i])$ for some $i$
    with prob $\mathbf{P}_{rep}$, $phrase_e, phrase_f \leftarrow phrase_e.append(phrase_e[i:]), phrase_f.append(phrase_f[i:])$ for
      some $i$
    return $word\_alignments$
```

scores of 97.84 and 98.12 respectively (Table 2). We consider en$_{\text{voice}}$ for our further experiments, as it has slightly better performance than the other one. To confirm the dependency of the model on spoken features, we check its behavior on similar spoken and written genre sentences. Figure 2 clearly shows the dependence of the model en$_{\text{voice}}$ on filler words and the use of the first person for the given example. A similar conclusion can also be drawn from Figure 3. It shows the last layer's mean attention scores for en$_{\text{voice}}$ model corresponding to [CLS] token. It can be observed that the tokens corresponding to *So*, *er*, *erm* have slightly higher attention scores than the others.

However, it fails to recognize repetition as a spoken feature. We suspect the lack of a significant amount of text with repetitions can be attributed to this behavior. The same can be observed for en$_{\text{base}}$ as well. Unlike the former model, en$_{\text{base}}$ also fails when non-lexical filler words are used. The fourth example in Figure 2 makes another interesting case. When the non-lexical filler *em erm* is replace with *umm*, the model fails to predict it correctly.

We have performance of hi$_{\text{base}}$ similar to en$_{\text{base}}$. Though it has the F1 score of 96.07%, it fairly depends ONLY on first person and lexical fillers.

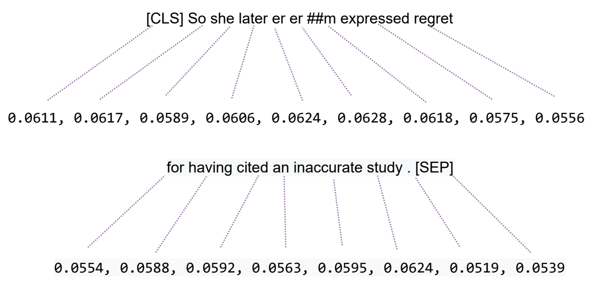


Figure 3: Attention scores of the last self-attention layer (for en$_{\text{voice}}$)

5 Rule-Based Injection of Spoken Language Features

The ultimate objective of our work is to improve translation quality of spoken language translation systems. However, spoken parallel corpora are rare, and the existing ones clearly lack spontaneity as evident from our previous discussion. In this work, we try to create spontaneous synthetic parallel corpus using the written corpora.

We propose a rule-based data augmentation method to add spoken features to the existing written examples on the phrase level (see Algorithm 1). The spoken texts tend to have informal words. They also have lesser content words with more grammatical words. Thus, we first back-translate

En (written): Krishnapuram is a village in Krishna district of the Indian state of Andhra Pradesh.
Hi (written): कृष्णारावुपालें कृष्णा में भारत के आन्ध्रप्रदेश राज्य के अन्तर्गत के कृष्णा जिले का एक गाँव है।
En (spoken_bt): Actually here's a village in ehh the Indian state ofland.
Hi (spoken_bt): वास्तव में अफगानिस्तान में भारतीय राज्य में भारत के भीतर भाग में एक गाँव है अ।
En (spoken): So Krishnapuram is a village in ehh Krishna district of the Indian state of Andhra Pradesh.
Hi (spoken): तो कृष्णारावुपालें कृष्णा में भारत के आन्ध्रप्रदेश राज्य के अन्तर्गत के कृष्णा जिले का एक गाँव है अ।

En (written): She later expressed regret for having cited an inaccurate study.
Hi (written): बाद में उसने एक गलत अध्ययन का हवाला देते हुए खेद व्यक्त किया।
En (spoken_bt): Later he err expressed regret by quoting a wrong study.
Hi (spoken_bt): बाद में उसने अ एक ग़लत अध्ययन को उद्धृत करने के द्वारा खेद व्यक्त किया ।
En (spoken): So she later expressed regret for having cited an inaccurate study.
Hi (spoken): तो बाद में उसने एक गलत अध्ययन का हवाला देते हुए खेद व्यक्त किया।

Figure 4: Two examples after applying algorithm 1. Highlighted sentences are of the written genre. *spoken_bt* denotes examples after applying algorithm on back-translated sentences, while *spoken* is without back-translation

the written genre parallel sentences to normalize them (Prabhumoye et al., 2018). We then create word alignments between En and Hi sentences using multilingual BERT (Devlin et al., 2018). We start with bigrams in the source language. Since we are interested in phrase alignments, we keep those bigrams with their indices of corresponding alignments belonging to a continuous span of numbers. We combine two bigrams and their corresponding alignments if they have overlapping indices. This ensures we get valid phrases and their mapping from En to Hi. We perform this combination process until convergence. After identifying phrases from the given word alignments, we add spoken features like non-lexical filler words (erm, emm, umm, eh, etc.), lexical filler words (so, like, etc.), and repetitions on both source and target sides with some probability P. We also add fillers at the beginning of sentences with probability p_i.

Figure 4 illustrates that back-translated outputs don't make enough sense. This can be attributed to the low quality of our En→Hi and Hi→En NMT models. Thus, we resort to using examples without back-translation. We denote this model by $\mathrm{M}_{\mathrm{rule}}$.

6 NMT with Genre Transfer

The previous method fails to transfer spoken features like informal words, and introducing more grammatical words. To counter this, we try data driven approaches. We stitch models trained on written and spoken genres together. We use this model to get translations across genres. We also try experiments with fine-tuning the stitched model on the augmented data created in Section 5.

We use pretrained NMT models by Tiedemann and Thottingal (2020) to fine-tune on our task. They train MarianMT (Junczys-Dowmunt et al., 2018) model on OPUS corpus (Tiedemann and Nygaard, 2004), and made the models publicly available. We fine-tune the models on the spoken and written genre datasets and label them as M_{sp} and M_{wr}, respectively, for both translation directions. We use Huggingface Trainer for our training purposes. We train the spoken and written models for 10 epochs with a batch size of 8. We evaluate at every 500 steps while using early stopping callback. During inference, we stitch M_{sp} and M_{wr} to get $\mathrm{M}_{\mathrm{mix}}$. We use the models to translate from En (and Hi) in written to Hi (and En) in spoken genre. We evaluate the generated parallel data on the written genre test set.

6.1 Model Stitching

During the training phase, for each translation direction, we train the encoder-decoder based NMT models on parallel data of styles wr (written) and sp (spoken). We get four models eventually: $M^{wr}_{en\to hi}$, $M^{sp}_{en\to hi}$, $M^{wr}_{hi\to en}$, and $M^{sp}_{hi\to en}$. We switch the encoders from sp models with wr models. We use the resulting model to translate text en of genre wr to hi of genre sp (and wr hi to sp en).

In general, each model M^j_i can be represented with $M^j_i(x) = f^{dec}_{i,j}(f^{enc}_{i,j}(x))$, for $i \in \{en \to hi, hi \to en\}$ and $j \in \{wr, sp\}$. We hypothesize that $f^{enc}_{i,j}$ encodes input x in a latent space independent of j. Thus, during inference (for x of style wr), to get translations of style sp, we can use $f^{dec}_{i,sp}$ with $f^{enc}_{i,wr}$.

6.2 Fine-tuning on Augmented Data

The previous method fails to give outputs in spontaneous spoken genre, as the existing training corpora have very few spontaneous examples. Thus, to get more natural-looking spontaneous spoken

Model	BLEU		(%) spoken	
	Hi-En	En-Hi	En	Hi
M_{wr}	34.31	35.11	2.33	3.19
M_{sp}	18.00	20.51	20.55	11.31
M_{mix}	19.27	20.88	20.98	12.32
M_{data_aug}	14.00	18.81	34.63	17.03
M_{rule}	46.93	76.09	21.70	15.05

Table 3: Overview of results. "% spoken" denotes proportion of generated outputs labelled as spoken.

	Content preservation		Fluency	
	Hi-En	En-Hi	Hi-En	En-Hi
M_{wr}	2.8	3.2	3.5	3.5
M_{sp}	1.7	2.5	3.0	3.4
M_{mix}	2.2	2.5	3.3	3.5
M_{data_aug}	2.0	2.2	2.9	3.2
M_{rule}	4.0	4.0	4.0	4.0

Table 4: Manual evaluation results

examples, we fine-tune our stitched model on the augmented data to get better spoken translations. We train it for 5 epochs while keeping other hyper-parameters the same as the other experiments.

7 Results

7.1 Genre Preservation

We evaluate the performance of our spoken data generation methods using automatic and manual evaluation. We use BLEU to check the quality of translations across genres and our BERT-based genre classifiers for En and Hi to get an estimated proportion of translated outputs in spoken genre. For manual evaluation, we randomly sample 50 outputs and evaluate them for content preservation (0-4; with 4: preserving all content, 2: one of context or nouns is preserved, 0: nothing preserved) and fluency (0-4; with 4 being most fluent).

Automatic and manual evaluation results can be found in Tables 3 and 4, resp.

M_{rule} outperforms other data-driven models in terms of BLEU, content preservation, and fluency scores. This is expected, as the input sentences stay the same apart from the injected spoken features. Among the data-driven approaches, M_{data_aug} deviates the most from the input sentences. The addition of explicit spoken features might be one of the significant reasons.

Interestingly, even though the spoken and written models are trained on a completely different dataset, without any overlap, the outputs generated from M_{mix} are quite good. M_{sp} has similar performance to M_{mix} for En→Hi direction. En verb forms, unlike Hi, are independent of the person. This can be one of the reasons for having a more generalized latent representation in the case of genre-specific NMT. Thus, switching encoders on the En side does not significantly affect the score.

Even though the classifier is biased towards specific spoken features, we can still use it to evaluate certain features. As expected, the genre score ("% spoken" in Table 3) for M_{wr} is pretty low. Although the scores for other models are relatively low, they are better than the written model. M_{data_aug} performs better than others in terms of the number of spoken sentences generated. The outputs look more naturally spoken than M_{rule} with less complex words and more grammatical words. However, it fails to align the filler words and repetitions in generated parallel data. This is obvious, as the models have not seen the positions of explicit spoken features in the other language. Also, since we are translating sentences across genres, the model seems to hallucinate while adding repetitions.

The model stitching method fails to inject even the lexical fillers it has seen during training. Out of 50 randomly sampled sentences, only 4 contained lexical features, contrary to the spoken training data, where almost 75% of the sampled examples contained such features. This is where M_{data_aug} and M_{rule} gets an advantage of controlling spoken features. Another issue with all data-driven approaches is the quality difference between Hi-En and En-Hi translation models. This affects the quality of parallel'ness of the augmented data.

Due to the difference in domains of written and spoken genres, the data-driven approaches struggle while handling proper nouns. This, along with unnecessary repeated addition of spoken features, results in a dip in performance of M_{data_aug}. This, however, can again be controlled by training it on augmented data with different frequencies of spoken features. The performance of M_{data_aug} improves with a decrease in the probability of the addition of spoken features. However, the alignment problem of such features still bothers the quality of generated parallel data. This is clearly handled during the rule-based approach, which makes it perform better than the other approaches.

7.2 Spoken Translation Quality

We check the utility of our spoken data generation methods via M_{sp}. We fine-tune M_{sp} further on 50k parallel data created using M_{mix}, M_{data_aug}, and our rule-based method M_{rule}. We label the models as SM_{M_Mix}, $SM_{M\_data\_aug}$, and SM_{data_aug} respec-

En: er so er i mean i i think erm you can be from er another country and i mean two different countries and then use a third language as linguafranca and then i think then it's a l- lingua franca
Hi (M_{sp}): मेरा मतलब है कि मुझे लगता है कि आप एक और देश से हो सकते हैं और मेरा मतलब है कि दो अलग-अलग देश और फिर एक तीसरी भाषा का उपयोग माफिया के रूप में करें और फिर मुझे लगता है कि यह एक ली-लिंगीका है।
Hi (SM_{data_aug}): तो अ मेरा मतलब है अम्म मुझे लगता है कि आप किसी दूसरे देश से अ हो सकते हैं और मेरा मतलब है कि दो अलग-अलग देश हैं और फिर लंगुफिया के रूप में एक तीसरी भाषा का उपयोग करें और फिर मुझे लगता है कि यह एक ल-हुआका है।

En: er i don't i don't know if a- a- am i right
Hi (M_{sp}): मैं नहीं जानता कि अगर एक-मैं सही हूँ-
Hi (SM_{data_aug}): अ मैं नहीं जानता कि मैं नहीं जानता कि क्या - मैं सही हूं

Figure 5: Two example outputs on VOICE dataset using M_{sp} and SM_{data_aug}

	hi-en	en-hi
M_{sp}	44.04	44.47
SM_{M_mix}	33.98	39.65
$SM_{M\_data\_aug}$	19.53	19.34
SM_{data_aug}	33.97	39.65

Table 5: BLEU scores of spoken translation systems on spoken test set from Samanantar (non-spontaneous)

	Cont. Pres.	Fluency	Ft_sc.
M_{sp}	2.7 ± 1.08	2.6 ± 1.07	0.14
SM_{M_mix}	1.9 ± 2.02	2.0 ± 2.11	0.14
$SM_{M\_data\_aug}$	2.6 ± 1.64	2.7 ± 1.76	0.28
SM_{data_aug}	**3.7** ± 0.42	**3.8** ± 0.42	**3.10**

Table 6: Manual evaluation results of En→Hi Spoken Translation System on VOICE 2.0. The results are calculated for 50 randomly sampled translated outputs.

tively. We first test the models on the test set of spoken parallel data from Samanantar using BLEU (Table 5). We also evaluate the En→Hi spoken translation system on VOICE 2.0 dataset manually (Table 6). Precisely, we use content preservation scores, fluency and spoken feature scores. We use spoken feature scores (Ft_sc.; +2 for fillers; +2 for repetitions) to check the quality of type and placement of filler words and repetitions.

Table 5 shows a drop in performance for models fine-tuned on augmented data on the spoken test set. The addition of sentences from new domain, along with updated syntactic structure seems to bring noise when compared with the spoken genre dataset. $SM_{M\_data\_aug}$ has the worst dip in the performance on the Samanantar test set. It tends to add extra filler words even if they are not present in the source sentence. This might be due to the misalignments of fillers and repetitions in the synthetic parallel sentences. The other two models SM_{data_aug} and SM_{M_mix} have comparable performance on the test set.

We check the En-Hi spoken translation systems quality on the VOICE dataset. SM_{data_aug} performs the best, and the placement of filler words and repetitions are also relatively accurate. The other three models struggled with such features. Figure 5 shows some translation outputs of SM_{data_aug}.

8 Discussion

The En→Hi spoken translation system fine-tuned on the synthesized data from our rule-based method performs quite well for spontaneous En examples. It not only applies the repetitions well, but also introduces the filler words at correct places. On the other hand, the baseline model failed to recognize and add fillers and repetitions. Since we didn't have any spontaneous examples in Hi, we could not empirically evaluate Hi→En. However, we expect similar performance from that as well.

We note that our output style evaluation relies on our genre classifier. For En, it has a varied training set but still it fails to work with unseen non-lexical fillers. For Hi, the classifier is even less reliable and should be used with caution. Clearly, there is a need for better genre classifiers.

It would be interesting to see the results for M_{mix} fine-tuned on back-translated data with added spoken features. Due to lower quality back-translation outputs, we could not perform the experiments with it. One approach can be finetuning the models on the Samanantar dataset for the Hi-En language pair and then using the model for back-translation. Another interesting experiment can also be checking the dependency of the model on controlled spoken features in training data and the extent to which they can be added without disturbing the content.

9 Conclusion

This paper proposes two main methods to synthesize parallel spoken data using existing written-genre parallel texts. Given the written input, the data-driven method produces a naturally-looking spoken output; however, it fails to ensure appropriate parallelism of explicit spoken features. The alternative rule-based approach has precise alignments of such features. Furthermore, we check the usability of the created parallel texts by fine-tuning Hi-En NMT models on merely 50k sentence pairs. The model fine-tuned on the synthetic corpus created using our rule-based method gives the best results. For En→Hi translation, it produces relatively decent results, and unlike the baseline model, it introduces correct non-lexical fillers at the proper places.

10 Acknowledgement

This work was supported by the GAČR grant Neural Representations in Multi-modal and Multilingual Modelling (NEUREM3, 19-26934X).

References

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*.

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.

Cicero dos Santos, Igor Melnyk, and Inkit Padhi. 2018. Fighting offensive language on social media with unsupervised text style transfer. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 189–194.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2016. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423.

Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. 2016. Generating topical poetry. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1191.

Harsh Jhamtani, Varun Gangal, Eduard Hovy, and Eric Nyberg. 2017. Shakespearizing modern language using copy-enriched sequence to sequence models. In *Proceedings of the Workshop on Stylistic Variation*, pages 10–19.

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.

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.

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*.

SM Lakew, A Erofeeva, M Negri, M Federico, and M Turchi. 2018. Transfer learning in multilingual neural machine translation with dynamic vocabulary. In *15th International Workshop on Spoken Language Translation*, pages 54–62.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Remi Mir, Bjarke Felbo, Nick Obradovich, and Iyad Rahwan. 2019. Evaluating style transfer for text. 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 495–504.

Xuan-Phi Nguyen, Shafiq Joty, Kui Wu, and Ai Ti Aw. 2020. Data diversification: A simple strategy for neural machine translation. *Advances in Neural Information Processing Systems*, 33:10018–10029.

Xing Niu, Marianna Martindale, and Marine Carpuat. 2017. A study of style in machine translation: Controlling the formality of machine translation output. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2814–2819, Copenhagen, Denmark. 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.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 866–876, Melbourne, Australia. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *54th Annual Meeting of the Association for Computational Linguistics*, pages 86–96. Association for Computational Linguistics (ACL).

Dinghan Shen, Mingzhi Zheng, Yelong Shen, Yanru Qu, and Weizhu Chen. 2020. A simple but tough-to-beat data augmentation approach for natural language understanding and generation. *arXiv preprint arXiv:2009.13818*.

Jörg Tiedemann and Lars Nygaard. 2004. The OPUS corpus - parallel and free: `http://logos.uio.no/opus`. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Jörg Tiedemann and Santhosh Thottingal. 2020. OPUS-MT — Building open translation services for the World. In *Proceedings of the 22nd Annual Conferenec of the European Association for Machine Translation (EAMT)*, Lisbon, Portugal.

Xuanxuan Wu, Jian Liu, Xinjie Li, Jinan Xu, Yufeng Chen, Yujie Zhang, and Hui Huang. 2021. Improving stylized neural machine translation with iterative dual knowledge transfer. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 3971–3977. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Ivan P. Yamshchikov, Viacheslav Shibaev, Aleksander Nagaev, Jürgen Jost, and Alexey Tikhonov. 2019. Decomposing textual information for style transfer. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 128–137, Hong Kong. Association for Computational Linguistics.

Rui Yan. 2016. i, poet: Automatic poetry composition through recurrent neural networks with iterative polishing schema. In *IJCAI*, volume 2238, page 2244.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232.

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.

PACMAN: PArallel CodeMixed dAta generatioN for POS tagging

Arindam Chatterjee[1,2]*, **Chhavi Sharma**[1]*, **Ayush Raj**[1], **Asif Ekbal**[2]
[1]Wipro AI Research Lab45, Bangalore, India
[2]Indian Institute of Technology, Patna, India
{arindam.chatterjee4, chhavi.sharma5, ayush.raj3}@wipro.com
asif@iitp.ac.in

Abstract

Code-mixing or Code-switching is the mixing of languages in the same context, predominantly observed in multilingual societies. The existing code-mixed datasets are small and primarily contain social media text that does not adhere to standard spelling and grammar. Computational models built on such data fail to generalise on unseen code-mixed data. To address the unavailability of quality code-mixed annotated datasets, we explore the combined task of *generating* annotated code-mixed data, and building *computational models* from this generated data, specifically for code-mixed Part-Of-Speech (POS) tagging. We introduce *PACMAN*[1] (*PA*rallel *C*ode*M*ixed d*A*ta generatio*N*) - a synthetically generated code-mixed POS tagged dataset, with above *50K* samples, which is the *largest annotated code-mixed dataset*. We build POS taggers using classical machine learning and deep learning based techniques on the generated data to report an F_1-score of 98% (8% above current State-of-the-art (SOTA)). To determine the efficacy of our data, we compare it against the existing benchmark in code-mixed POS tagging. PACMAN outperforms the benchmark, ratifying that our dataset and, subsequently, our POS tagging models are generalised and capable of handling even natural code-mixed and monolingual data.

1 Introduction

Code-Mixing or Code-Switching is primarily observed and archetypal in multilingual societies across the globe (Gumperz, 1964; Thompson, 2009; Auer, 2020; Schwab, 2021). Although linguists differentiate between code-mixing (Boggs, 1983) and code-switching (Myers-Scotton, 1993), we use CM to mean both. CM has recently garnered significant interest for researchers because of its gradual emergence as the *prima lingua* for social media posts, blogs, chats, and messages.

The first computational work on CM text was explored by Solorio and Liu (2008) for English-Spanish POS tagging. They used individual POS taggers for English and Spanish and heuristics, machine learning, and word-level language information (WLI) to find the optimal tag for each word. Subsequent work on POS Tagging for CM text like Jamatia et al. (2015); Vyas et al. (2014) use similar approaches, using language information and a combination tagger on *social media* CM text. They emphasize that WLI is a mandatory ingredient for CM POS tagging, necessitating the use of a combination tagger. Recent work by Singh et al. (2018) applies a set of hand-crafted features (including WLI) to POS tag CM text, with higher accuracy, and *without* using a combination tagger. This establishes that CM computational models can be *independently* built without using models from constituent languages.

Code-mixing is abundantly observed in our daily lives through personal chats, messages *etc.*, apart from social media. Since code-mixing is majorly used on social media, CM datasets are built primarily on social media data, which is *domain-specific*, *noisy*, and has *non-standard spellings* and *grammar*, especially for languages written in scripts other than Roman (Vyas et al., 2014). This makes *annotation* of such data a separate challenge in itself. Additionally, extracting genuine CM data from social media is a difficult task, accounting for: **(i)** inadequate code-mixed datasets (Jose et al., 2020), **(ii)** small sizes of available CM datasets and **(iii)** low Code-mixing Ratio (CMR) (discussed in Section 2) of such datasets (Jamatia et al., 2015; Singh et al., 2018). To address these existing bottlenecks of CM language research, through this work, we explore the following fundamental question:

Can we generate high-quality annotated CM data and build computational models with it, comparable to natural CM data and its subsequent computational models?

* denotes equal contribution
[1]PACMAN dataset to be released later

To address the above question, we explore the dual task of annotated CM data *generation* and building *computational models* from this generated data, specifically for CM POS tagging. We introduce **PACMAN** (**PA**rallel **C**ode**M**ixed d**A**ta generatio**N**) for CM POS tagging. Through **PACMAN**, our motivation is to address the gaps in the existing CM datasets and computational models.

First, we generate annotated CM POS tagged data for *Hinglish*, through an *alignment*, *annotation* and *replacement* strategy from parallel *Hindi-English* corpus. The generated data has standard spelling and grammar, unlike social media datasets. Further, this alleviates the strenuous and difficult task of annotating social media CM data. CM datasets usually have sizes equal to or less than a meagre *1000* samples (Jose et al., 2020). We report a corpus of above ***50K*** annotated POS tagged CM samples, which are 100% code-mixed. Although our data is generated for *Hinglish*, we claim that our technique can be used to generate annotated data for any CM language pair, provided a parallel corpus and POS taggers are available for both languages. To our knowledge, our work is a *first-of-a-kind*, with respect to the generation of CM annotated data. We discuss our data generation pipeline in detail in Section 3.

Second, we build POS taggers using both Machine Learning (ML) and Deep Learning (DL) based techniques on PACMAN data. Singh et al. (2018) reports better accuracy with ML techniques than DL models. Through this work, we illustrate that this is predominantly due to the *inadequacy* of data, typically required to build better deep learning models. Our DL model outperforms our ML model by 1.5%. We would like to assert here that we build our models independently on CM data itself and *not* using a *combination tagging strategy*. We also analyse the contribution of WLI on the CM POS tagging task by infusing language information into the models. The results obtained reveal that with sizeable data, instances where language information is useful, dwindle to a small fraction, as we obtain similar accuracy without language information. We discuss our POS taggers in detail in Section 4.

Third, in order to gauge the quality of our data and models, we test them against the existing benchmark in CM POS tagging (Aguilar et al., 2020), which is based on social media text. Our models outperform the SOTA benchmark by 10%, despite having a *higher* CMR. Our results also show that our models built on PACMAN data are able to handle social media text as well as monolingual text. This is largely because of the size of our dataset, standard spelling and grammar integral to PACMAN, and uniform distribution of words and POS tags across Hindi and English. This makes our models more generalized and equipped to handle different types of CM data. We discuss qualitative analysis of our data in Section 5, error analysis in Section 6 and conclude in Section 7.

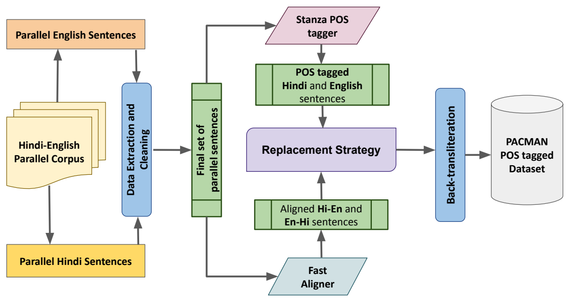
2 Code-mixing Terminology

In this section we define a few terms akin to code-mixing, that we use in later sections of the paper:

- **Matrix Language (MtxL) and Embedded Language (EL):** In a code-mixed sentence, containing two (2) languages, the *base* language, or the language from which the sentence is 'coming from' is called the *matrix language* and the other language is the *embedded language* (Joshi, 1982)
- **Switching Point (SP):** Switching Points are the junctions in a code-mixed text, where the language switches (Chatterjee et al., 2020).
- **Code-Mixing Index (CMI):** It is the measurement of the level of mixing between the constituent languages in a code-mixed context (Gambäck, 2014).
- **Code-Mixing Ratio (CMR):** CMR is the fraction of samples in a code-mixed corpus that are actually code-mixed.

3 Data Generation Pipeline

Code-mixed language research has recently received a lot of attention in the NLP community. Despite the recent interest, Jose et al. (2020) report that there is a scarcity of datasets in the domain, and existing datasets are very small (*1000* samples per dataset on average). Besides, as CM is prevalent on social media, the existing CM datasets contain only social media text, which have non-standard spelling and grammar and are difficult to annotate. In order to bridge this gap, in the present work, we endeavour to generate an *exhaustive corpus* of *annotated code-mixed data*, that is generic across domains and has standardised spelling and grammar, unlike social media data.



Parallel English Sentences
Hindi-English Parallel Corpus
Parallel Hindi Sentences
Data Extraction and Cleaning
Final set of parallel sentences
Stanza POS tagger
POS tagged Hindi and English sentences
Replacement Strategy
Aligned Hi-En and En-Hi sentences
Fast Aligner
Back-transliteration
PACMAN POS tagged Dataset

Figure 1: Data generation pipeline for PACMAN

We introduce **PACMAN** (**PA**rallel **C**ode**M**ixed d**A**ta generatio**N** a *one-of-a-kind* data generation framework for CM **POS** tagging. It is a synthetically generated data framework, curated via an *alignment*, *annotation* and *replacement* based strategy. Our approach is similar to the one applied by Srivastava and Singh (2021), but with the added objective of *uniform distribution of words* across constituent languages of the CM language pair. Currently, PACMAN is implemented for *Hinglish*, but using the same strategy will work for *any* CM language pair. The subsequent subsections present the details involved in creating the PACMAN dataset.

3.1 Resources and Tools

As already discussed, we generate the PACMAN dataset using an align, annotate, and replace strategy. In order to facilitate this, we use the following resources and tools:

- **Parallel Corpus**: We have used the *Hindi-English* parallel corpus proposed by Kunchukuttan et al. (2018) containing 1.6M English-Hindi parallel sentences. The Hindi counterpart of the corpus is available in *Devanāgari* script.

- **Fast Align**: It is a fast, simple and unsupervised aligner introduced by Dyer et al. (2013) that learns the word alignment between the parallel sentences using log-linear reparameterisation of IBM Model 2, based on a variation of the lexical translation models proposed by Brown et al. (1993).

- **Stanza**: An open-source python NLP toolkit proposed by Qi et al. (2020), that covers a wide range of text analytics tools, such as tokenization, lemmatization, POS tagging *etc.* in 66 languages.

- **Indic-trans**: It is used for transliteration framework proposed by Bhat et al. (2014) to convert the *Devanāgari* script to *Roman* and vice-versa, based on a structured perceptron model that uses letter alignments learned from GIZA++ (Och and Ney, 2003).

3.2 Methodology

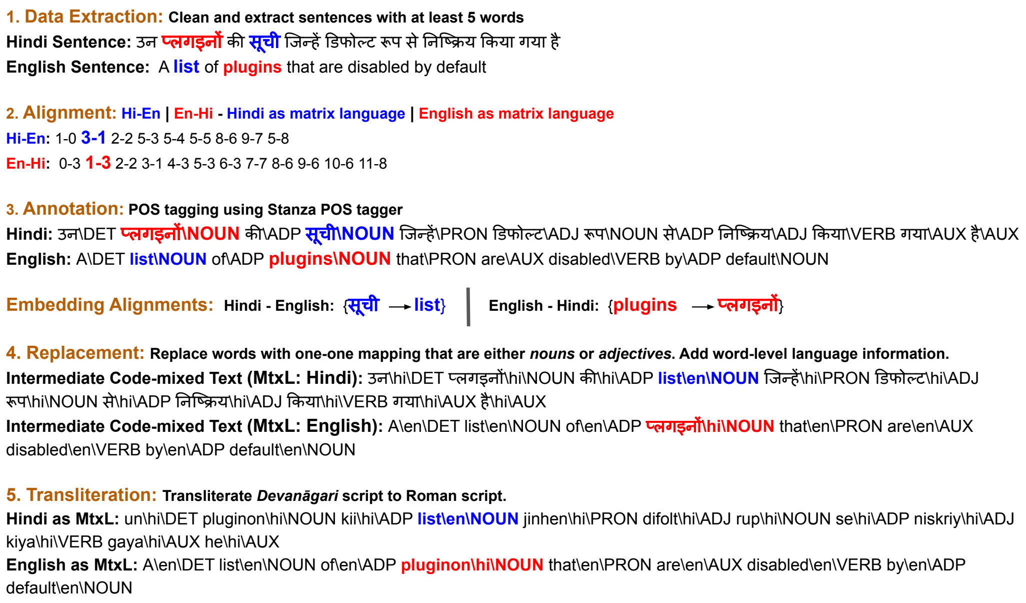
For generating annotated POS tagged *Hinglish* data, we use a *Hindi-English parallel corpus* and combine parallel sentences following the *matrix language theory* introduced by Joshi (1982), using an elegant rule-based algorithm illustrated in Figure 1. A similar strategy was proposed by Srivastava and Singh (2021), but they only embed English words in Hindi sentences (Hindi as *matrix language*). We additionally embed Hindi words in English sentences (English as *matrix language*). This ensures a *uniform distribution* of words and subsequently POS tags across both the constituent languages *viz.*, Hindi and English. These word replacements account for the introduction of switching points and hence code-mixing of the generated sentences. Analysis of code-mixed sentences reveals that the words where the switch in languages happens, are primarily *nouns* and in some cases *adjectives* (Srivastava and Singh, 2021). We discuss each step of our data generation pipeline in detail below:

1. **Data Extraction:** From the parallel corpus, we filter out sentences of either language with less than *five* (5) words and instances where the Hindi parallel sentence contains English words. We pre-process and clean the resulting samples by removing unimportant tokens like URLs *etc.*

2. **Alignment:** Next, we train *fast aligner* on the pre-processed sample, which provides aligned indices of words for Hindi to English as well as English to Hindi, for each parallel sentence.

3. **Annotation:** Once the alignment is done, we POS tag both the Hindi and English sentences using *Stanza* POS tagger. We use the *universal* POS tagset (discussed in Section 4.1), which is available as part of Stanza to annotate the parallel sentences.

4. **Replacement:** This step is the most critical step in our strategy. Once the Hindi and English parallel sentences are aligned and POS tagged, we look for *one-one* word mappings in


1. Data Extraction: **Clean and extract sentences with at least 5 words**
Hindi Sentence: उन **प्लगइनों** की **सूची** जिन्हें डिफोल्ट रूप से निष्क्रिय किया गया है
English Sentence: A **list** of **plugins** that are disabled by default

2. Alignment: **Hi-En** | **En-Hi** - **Hindi as matrix language** | **English as matrix language**
Hi-En: 1-0 **3-1** 2-2 5-3 5-4 5-5 8-6 9-7 5-8
En-Hi: 0-3 **1-3** 2-2 3-1 4-3 5-3 6-3 7-7 8-6 9-6 10-6 11-8

3. Annotation: **POS tagging using Stanza POS tagger**
Hindi: उन\DET **प्लगइनों\NOUN** की\ADP **सूची\NOUN** जिन्हें\PRON डिफोल्ट\ADJ रूप\NOUN से\ADP निष्क्रिय\ADJ किया\VERB गया\AUX है\AUX
English: A\DET **list\NOUN** of\ADP **plugins\NOUN** that\PRON are\AUX disabled\VERB by\ADP default\NOUN

Embedding Alignments: **Hindi - English:** {सूची → list} | **English - Hindi:** {plugins → प्लगइनों}

4. Replacement: **Replace words with one-one mapping that are either *nouns* or *adjectives*. Add word-level language information.**
Intermediate Code-mixed Text (MtxL: Hindi): उन\hi\DET प्लगइनों\hi\NOUN की\hi\ADP **list\en\NOUN** जिन्हें\hi\PRON डिफोल्ट\hi\ADJ रूप\hi\NOUN से\hi\ADP निष्क्रिय\hi\ADJ किया\hi\VERB गया\hi\AUX है\hi\AUX
Intermediate Code-mixed Text (MtxL: English): A\en\DET list\en\NOUN of\en\ADP **प्लगइनों\hi\NOUN** that\en\PRON are\en\AUX disabled\en\VERB by\en\ADP default\en\NOUN

5. Transliteration: **Transliterate *Devanāgari* script to Roman script.**
Hindi as MtxL: un\hi\DET pluginon\hi\NOUN kii\hi\ADP **list\en\NOUN** jinhen\hi\PRON difolt\hi\ADJ rup\hi\NOUN se\hi\ADP niskriy\hi\ADJ kiya\hi\VERB gaya\hi\AUX he\hi\AUX
English as MtxL: A\en\DET list\en\NOUN of\en\ADP **pluginon\hi\NOUN** that\en\PRON are\en\AUX disabled\en\VERB by\en\ADP default\en\NOUN

Figure 2: A sample execution of our data generation pipeline on example parallel sentences. A sample token in PACMAN is of the form: `<word/LID/POS>`. We have used the *universal* POS tagset. For the language identifier (LID), each token is tagged as ***hi*** (Hindi), ***en*** (English) or ***rest*** (others). The final generated text for PACMAN data both for *Hindi* and *English* as *matrix languages* can be observed.

the alignment that are either a *noun* or an *adjective*. If such a mapping exists, from either Hindi to English or English to Hindi alignments, we call them *embedding alignments*. Essentially, we locate junctions in the parallel sentences where code-mixing can happen. For the Hindi parallel sentence, we replace the words in Hindi with the words in English that constitute the *embedding alignments*. This is the case where Hindi is the *matrix language*, and English is the *embedded language*. The same is actuated for English as the MtxL and Hindi as the EL. Using both Hindi and English as matrix languages ensures that the vocabulary as well as the POS tag distribution is uniform across both languages. We add the word-level language information (*hi* for Hindi, *en* for English, *rest* for others) in the generated sentences, as the information is known based on the *matrix* and *embedded* language.

5. **Transliteration:** The Hindi words in the generated data, are in *Devanāgari* script, as they occur in the same form in the parallel corpus. We use *Indic-trans* to transliterate the words in *Devanāgari* to Roman script in order to generate the final PACMAN code-mixed annotated data, which is *completely* in Roman script.

The generated PACMAN dataset is Roman in form and each word (token) is annotated with a language identifier (LID) (*hi* for Hindi, *en* for English, *rest* for others) and POS label from Stanza. A sample token of PACMAN is of the form: `<word/LID/POS>`. We have demonstrated the execution of our data generation pipeline for a sample set of parallel sentences in Figure 2. An example of generated PACMAN data can also be visualized in this figure.

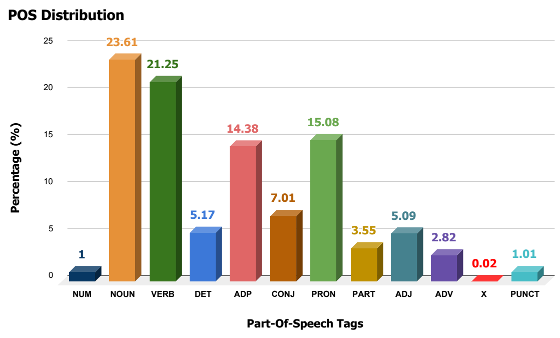


Figure 3: Percentage-wise POS distribution of PACMAN over 12 universal POS tags

3.3 Corpus Statistics

We generate the PACMAN dataset through an *alignment*, *annotation* and *replacement* based strategy discussed in Section 3.2. The generated dataset contains **51118** unique pure CM *Hinglish* POS tagged sentences, with an average CMI of **14.61** and **100%** CMR. The average sample length in the dataset is **28** as shown in Table 2. The POS distribution over 12 universal POS tags is shown in Figure 3.

4 Part-Of-Speech Tagging

In this section, we discuss the tagset used for POS annotation, details and performance of the sequence labeling models applied on the PACMAN dataset for word-level POS prediction. In addition, we also analyse the contribution of WLI for the CM POS labeling task.

4.1 Part-Of-Speech tagset

We have already discussed earlier that during our data generation phase, we use the *Universal POS tagset* introduced by Petrov et al. (2012), who established that a coarse-grained POS tagset of **12** tags is sufficient for POS tagging and performs well for the task, compared to a fine-grained tagset as in Marcus et al. (1993). The *Stanza* POS tagger has provision for the universal POS tagset (called *upos*), but has *17* unique POS tags. In order to maintain the 12 POS tags proposed by Petrov et al. (2012), we have post-processed our data by replacing `AUX` to `VERB`, `PROPN` to `NOUN`, `SCONJ` and `CCONJ` to `CONJ`. Further, from the POS distribution statistics shared in Figure 3, it can be seen that tag `X` contributes only **0.02%** of total tag counts. Hence, we remove the samples containing `X` tag from our dataset, as it is statistically insignificant. The final set of **11** POS tags are: `ADJ, ADP, ADV, VERB, CONJ, DET, NOUN, NUM, PART, PRON, PUNCT`.

4.2 Sequence Labeling Models

To establish the efficacy of the generated PACMAN dataset, we built sequence labeling models on it. We experimented with an ML model and a DL model *viz.*, Conditional Random Field (CRF) and Bidirectional LSTM (BiLSTM). Previous research has validated the use of CRFs (Toutanova et al., 2003; Choi et al., 2005; Peng and McCallum, 2006) and LSTMs / BiLSTMs (Ghosh et al., 2016; Wang et al., 2015) for POS tagging and other sequence labeling NLP tasks. We experimented with the transformer-based models as in Aguilar et al. (2020), but do not report them, as they were outperformed by CRF and BiLSTM.

We have used the CRF model proposed by Lafferty et al. (2001), using the faster ”*L-BFGS*” (Liu and Nocedal, 1989) optimization. We used the following set of hand-crafted linguistic features for the CRF classifier: **(i)** The current token W, **(ii)** index of the W, **(iii)** affixes of length 1 to 3, **(iv)** a binary feature indicating whether all characters in W are uppercase or lowercase, **(v)** a binary feature indicating whether W has any upper case character, **(vi)** a binary feature indicating whether there is any digit character in W, **(vii)** previous and next word of W, **(viii)** a binary feature indicating whether W has a hyphen (-). To prevent over-fitting, we use L_1 and L_2 regularization. We used grid search to extract the optimal hyper-parameters for the CRF model. We call this model PACMAN$_{CRF}$.

We do not use any hand-crafted features for our BiLSTM model. Instead, we train a set of *word embeddings* as part of the neural network designed for the word level POS prediction task. This ensures that word embeddings are tuned for the POS tagging task. We have kept the dimension of the word embeddings as *128*. These embeddings are passed on to the BiLSTM layer (output dimension *512*), followed by a set of *feed forward* layers (dimensions *512* and *256*), and finally a *softmax layer* for the POS prediction. To prevent over-fitting, we add a dropout (*0.25*) layer and L_1, L_2 regularizations. We experimented with different sets of hyper-parameters, layer sequences, and dimensions, but this configuration yielded the best performance. We call this model PACMAN$_{BiLSTM}$.

For both the sequence labeling tasks, we take a **75:5:20** split for *training*, *validation* and *testing* sets for our models. The *Precision*, *Recall* and *F$_1$-score* for PACMAN$_{CRF}$ and PACMAN$_{BiLSTM}$ are reported in the first half of Table 1. It can be observed that the PACMAN$_{CRF}$ model achieves an overall F$_1$-score of **0.965**, whereas the PACMAN$_{BiLSTM}$ model outperforms the PACMAN$_{CRF}$ model with an overall F$_1$-score of **0.979**. Singh et al. (2018) reports that ML-based techniques work better than DL-based techniques for CM POS labeling. Our results show that this is predominantly due to the *inadequacy of data*, typically required to build better deep learning models. Since, our data is almost *50* times that of Singh et al. (2018), our DL model *viz.*, PACMAN$_{BiLSTM}$ with an F$_1$-score of 98% performs better than our ML model PACMAN$_{CRF}$ by 1.5%.

POS	Without WLI						With WLI					
	PACMAN$_{CRF}$			PACMAN$_{BiLSTM}$			PACMAN$^{L}_{CRF}$			PACMAN$^{L}_{BiLSTM}$		
	P	R	F	P	R	F	P	R	F	P	R	F
ADJ	0.939	0.92	0.929	0.955	0.946	0.95	0.943	0.912	0.927	0.95	0.948	0.949
ADP	0.983	0.987	0.985	0.988	0.99	0.989	0.985	0.989	0.987	0.988	0.991	0.99
ADV	0.914	0.908	0.911	0.947	0.956	0.952	0.915	0.902	0.909	0.951	0.956	0.954
CONJ	0.959	0.968	0.963	0.978	0.976	0.977	0.959	0.97	0.964	0.979	0.975	0.977
DET	0.972	0.956	0.964	0.988	0.984	0.986	0.978	0.962	0.97	0.99	0.983	0.987
NOUN	0.974	0.975	0.974	0.983	0.981	0.982	0.97	0.977	0.974	0.981	0.983	0.982
NUM	0.963	0.978	0.97	0.974	0.984	0.979	0.993	0.988	0.991	0.99	0.987	0.988
PART	0.987	0.989	0.988	0.994	0.993	0.993	0.99	0.99	0.99	0.993	0.992	0.993
PRON	0.975	0.978	0.976	0.99	0.991	0.99	0.978	0.98	0.979	0.99	0.993	0.991
VERB	0.971	0.973	0.972	0.988	0.99	0.989	0.975	0.976	0.975	0.99	0.988	0.989
PUNCT	0.983	0.983	0.983	0.992	0.981	0.986	0.985	0.983	0.984	0.99	0.983	0.986
Avg	0.965	0.965	**0.965**	0.98	0.979	**0.979**	0.97	0.966	**0.968**	0.981	0.98	**0.981**

Table 1: Precision (P), Recall (R) and F$_1$-score (F) for CRF and BiLSTM sequence labeling models, on PACMAN data, with and without WLI. It can be observed that the PACMAN$_{BiLSTM}$ performs better than the PACMAN$_{CRF}$. Also, infusing WLI parameter *does not* boost the F$_1$-scores for both models.

4.3 Contribution of Word-Level Language Information

Solorio and Liu (2008); Vyas et al. (2014) emphasize that word-level language information (WLI) is a requisite for POS tagging CM text. Singh et al. (2018) have shown a slight increase in the overall F$_1$-score (2%) when language information is considered. In Section 4.2, we observed the accuracy of our sequence labeling models on CM POS tagging *without* language information for each word. To gauge the effect of language information on CM POS tagging, we model the POS sequence labeling task with the *annotated* WLI, which is captured during data generation. We *do not* build a *language identifier* (LID) model. The WLI tags are added to the train, validation, and test data, split in **75:5:20**, as discussed in Section 4.2. For the PACMAN$_{CRF}$ model, we add a *language feature* for word W, and for PACMAN$_{BiLSTM}$, we model the language information for each word, as part of the *word embeddings*. We name the models PACMAN$^{L}_{CRF}$ and PACMAN$^{L}_{BiLSTM}$ respectively.

The *Precision*, *Recall* and *F$_1$-score* for PACMAN$^{L}_{CRF}$ and PACMAN$^{L}_{BiLSTM}$ are reported in the second half of Table 1. It is evident from the results that post the infusion of WLI, there is almost no increase in performance (0.3% for CRF and 0.2% for BiLSTM) of the CM POS label prediction, even though *ground-truth* WLI labels are provided in test data and **not** *probabilistic* labels from a LID model. We also found that the predicted POS labels for PACMAN$_{BiLSTM}$ and PACMAN$^{L}_{BiLSTM}$ differ by only 1.13%. As our findings are contrary to previous research in CM POS tagging, we investigated the statistical significance of our findings. We obtained a *p-value* of $9.6e^{-9}$ ($<< 0.05$ threshold), validating that our observations are *statistically significant*.

This establishes that for a significantly *large dataset*, that is *uniformly distributed* across constituent languages of a CM setting, the models learn the sequential data better and are able to assign classes to each word based on the context, *inherently* capturing the language information for each word. Thus, the requirement of WLI is *nullified* for the CM POS labeling task.

It can be observed in Table 1 that despite similar overall F$_1$-scores, there is a dissimilarity in Precision, Recall, and F$_1$-scores of individual POS tags. To understand this, we did further analysis on the impact of WLI on our sequential models. We found that WLI aids in the correct POS category identification of words when they have the *exact same spelling* in both English and transliterated Hindi. Words like `the` (was), `he` (is), `main` (me) which are also English words, are some examples of such cases (example shown in Appendix A through Figure 5). These cases are just a *handful* though, and hence ***do not*** affect the overall accuracy of the sequence labeling models.

5 PACMAN: Qualitative Evaluation

In this section, we gauge the quality of our dataset and models against the existing *benchmark* for CM POS tagging *i.e.*, *Linguistic Code-switching Evaluation* (LinCE), reported by Aguilar et al. (2020). Since PACMAN is synthetically generated, it is crucial that its efficacy is tested against a benchmark social media dataset (LinCE), which is considered natural CM data. To this end, we devise a set of intricately designed experimental scenarios for this comparative investigation.

5.1 Dataset Statistics Comparison

We first compare the statistical parameters across the proposed PACMAN dataset and the benchmark dataset LinCE, exhibited in Table 2. The comparison of POS distributions between PACMAN and LinCE are also shown in Appendix A through Figure 4.

Parameters	PACMAN	LinCE
# Code-mixed samples	**51118**	**1077**
# English samples	0	343
# Hindi samples	0	69
CMR (%)	**100**	**72.33**
Average CMI	**14.12**	**14.16**
Avg. Sample Length	28.16	15.11
Total # tags	1477765	26416

Table 2: Comparison of statistics between PACMAN and LinCE code-mixed datasets. The key parameters to note here are the avg CMI, CMR and avg sample length.

In terms of the number of samples and tags, PACMAN is almost *50* times that of LinCE. Jamatia et al. (2015) stated that in order to compare, two CM datasets, it is imperative that their complexities are similar *i.e.*, their average CMIs are close to each other. The average CMIs of both PACMAN and LinCE are around **14** and thus comparable. As for the number of actual CM samples, PACMAN and LinCE have CMR values of 100% and 72%, respectively. The average sample length in LinCE is *15* and that of PACMAN is *28*.

5.2 Experimental Setup

Comparing code-mixed datasets is tough, due to differences in source and level of mixing observed across such datasets. The benchmark LinCE dataset is based on *tweets*, extracted over a handful of *topics*, making it domain-specific and noisy in nature. Whereas PACMAN is domain agnostic and has standard spelling and grammar. We pre-processed both PACMAN and LinCE, for homogeneity, and built a set of customised scenarios to compare them.

5.2.1 Data Pre-processing

For the LinCE dataset, we observed that the authors had followed a customised annotation scheme, with *3* extra tags *viz.*, `PART_NEG`, `PRON_WH` and `PROPN`. To map LinCE to the standard universal POS tagset, we converted `PART_NEG` to `PART`, `PRON_WH` to `PRON` and `PROPN` to `NOUN`. Further, the LinCE dataset does not contain the `PUNCT` tag and PACMAN does not contain `X` tag. We remove all occurrences of `PUNCT` from PACMAN and `X` from LinCE, without affecting the context/ meaning of the samples, resulting in a total of 10 POS tags across both datasets. We name these pre-processed datasets PACMAN$_{PCD}$ and LinCE$_{PCD}$.

5.2.2 Comparison Scenarios

We gauge the quality of the PACMAN dataset against the LinCE benchmark, without language information, as we have already established in Section 4.3 that WLI does not affect the accuracy of the sequence labeling models. To compare the pre-processed datasets (PACMAN$_{PCD}$ and LinCE$_{PCD}$), we devised a set of experimental scenarios. For actuating the best performance framework, we utilise the highest performing models for LinCE (**CRF** which is the SOTA, as reported by Singh et al. (2018)), and PACMAN (**BiLSTM**) as discussed in Section 4) on the following carefully devised experimental scenarios:

1. S$_1$: Trained and tested on LinCE data.
   Train: LinCE$_{PCD}$ | **Test:** LinCE$_{PCD}$
   Purpose: Estimate benchmark accuracy

2. S$_2$: Trained and tested on PACMAN data.
   Train: PACMAN$_{PCD}$ | **Test:** PACMAN$_{PCD}$
   Purpose: Measure PACMAN accuracy

3. S$_3$: Trained on LinCE, tested on PACMAN.
   Train: LinCE$_{PCD}$ | **Test:** PACMAN$_{PCD}$
   Purpose: Evaluate how well LinCE generalises on PACMAN data (standard spelling and grammar)

4. S$_4$: Trained on PACMAN, tested on LinCE.
   Train: PACMAN$_{PCD}$ | **Test:** LinCE$_{PCD}$
   Purpose: Gauge how well PACMAN generalises on LinCE data (social media text)

5.3 Results and Discussion

The results of the experiments proposed in Section 5.2.2 are shown in Table 3. For scenarios S$_1$ and S$_2$, it can be seen that PACMAN outperforms the benchmark LinCE, by 9% and 14% for CRF and BiLSTM, respectively. Essentially, PACMAN$_{BiLSTM}$ (S$_2$: BiLSTM, with an f$_1$-score of 98%) surpasses the SOTA benchmark (S$_2$: CRF, with an f$_1$-score of 88%) by 10%.

Comparing the scenarios S$_3$ and S$_4$, it can be seen that the models that are trained on PACMAN data perform better than the ones trained on LinCE data (7% and 10% for CRF and BiLSTM respectively). Also, CRF outperforms BiLSTM with LinCE data, as LinCE dataset is small and not adequate for training/testing the BiLSTM model (as discussed in Section 4.2).

POS	CRF				BiLSTM			
	S_1	S_2	S_3	S_4	S_1	S_2	S_3	S_4
ADJ	0.67	0.93	0.34	0.57	0.65	0.95	0.27	0.50
ADP	0.94	0.98	0.85	0.79	0.93	0.99	0.86	0.83
ADV	0.82	0.91	0.49	0.62	0.80	0.98	0.48	0.76
CONJ	0.91	0.96	0.79	0.58	0.92	0.99	0.79	0.75
DET	0.88	0.96	0.64	0.80	0.87	0.99	0.68	0.78
NOUN	0.88	0.97	0.72	0.79	0.81	0.98	0.41	0.72
NUM	0.96	0.97	0.76	0.86	0.92	0.94	0.70	0.59
PART	0.84	0.99	0.54	0.75	0.86	0.99	0.64	0.74
PRON	0.86	0.98	0.67	0.72	0.83	0.98	0.58	0.66
VERB	0.90	0.98	0.72	0.83	0.82	0.98	0.55	0.62
Average	0.88	**0.97**	0.70	**0.77**	0.84	**0.98**	0.60	**0.70**

Table 3: F_1-scores obtained for CRF and BiLSTM on scenarios S_{1-4}. S_1 and S_2 show that PACMAN outperforms the benchmark by 9%. S_3 and S_4 show that PACMAN generalises better on LinCE data, but LinCE is not able to generalise equally well on PACMAN data.

This essentially emphasizes that: **(i)** Sequence labeling models trained on PACMAN data are able to *generalise better* than LinCE, even on social media data, as PACMAN data follows standard spelling and grammar. **(ii)** Models trained on LinCE (social media data) do not generalise as well on standardised data. **(iii)** LinCE has 28% monolingual data, but PACMAN is able to handle monolingual data as well, due to the *exhaustive* and *uniform* word and POS tag distribution across *Hindi* and *English*. We can conclusively establish that not only is PACMAN *larger* and *standardised*, but also computationally *superior*, and is capable of *generalising* on natural CM data as well as monolingual data.

6 PACMAN: Error Analysis

With the motivation of bridging gaps in existing CM datasets and computational models we generate PACMAN, using an *alignment*, *annotation* and *replacement* strategy from parallel *Hindi-English* corpus. Although the generated data is *clean*, *formalised*, and yields *impressive* results, our analysis shows that the usage of *probabilistic* tools and resources adversely affects the quality of the generated dataset, although such cases are *scarce*. We highlight these inaccuracies with examples, for every *stage* of our data generation pipeline.

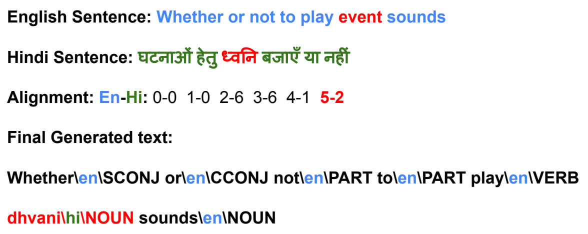
Inaccuracies in the Parallel Corpus: We observed a few errors in the Hindi parallel sentences in the corpus like:

1. Multiple *Devanāgari* forms of same Hindi words
   e.g., है/ हे - *hai* (is); नहि/ नहिं - *nahi* (no)

2. Hindi translations of English words being merely their *Devanāgari* forms
   e.g., डिफोल्ट (default) , डेटाबेस (database)

This results in *incorrect non-standard* words in the dataset, from the transliterations (*he* (hai), *nahin* (nahi), *difolt* (default), *databes* (database)).

Inaccurate Alignment: Due to inherent constraints in Fast Align, sometimes the alignment results rendered are inaccurate. This leads to incorrect word replacements and *redundant* words, resulting in erroneous data. In the example below it can be observed that the word *dhvani* (sound) is redundant and incorrect, due to the inaccurate alignment of the words ध्वनि(dhvani) and *event*.



English Sentence: Whether or not to play event sounds

Hindi Sentence: घटनाओं हेतु ध्वनि बजाएँ या नहीं

Alignment: **En-Hi:** 0-0 1-0 2-6 3-6 4-1 **5-2**

Final Generated text:

Whether\en\SCONJ or\en\CCONJ not\en\PART to\en\PART play\en\VERB dhvani\hi\NOUN sounds\en\NOUN

Inaccurate POS tagging: Due to a more flexible grammatical structure in Hindi, Stanza is occasionally unable to correctly adjudicate the relative positions of *adjectives* and *adverbs w.r.t nouns* and *verbs*. For *e.g.*, ADJ: बुरी / *buri* (bad) can come in one of the following orders:

1. *Before* a NOUN: बुरी चीज / *buri cheej* (bad thing)

2. *After* a NOUN: आदत बुरी / *aadat buri* (bad habit)

Stanza annotates the latter as: *aadat*\NOUN *buri*\NOUN, due to the confusion infused by Hindi grammar, resulting in an error in the dataset.

Stanza is capable of handling multi-word expressions (MWE), which leads to poor tagging resolutions in some cases. For *e.g.*, for the word sequence *ambient light* is detected as a MWE, resulting in *ambient* being annotated as NOUN instead of ADJ. This results in the incorrect entry '*ambient*\en\ADJ *prakaash*\hi\NOUN' in the dataset.

Inaccurate Transliteration: In some cases, Indic-trans renders erroneous transliteration. For *e.g.*, 'ऐ' (hey) in 'ऐ राम इधर आओ' (Hey Ram come here), is transliterated to 'I', which leads to Stanza *incorrectly* tagging 'ऐ' as `PRON`, instead of `PART`, resulting in inefficient data.

7 Conclusion and Future-work

In this paper, we address some of the existing limitations in the datasets and computational models for code-mixed languages, specifically for the CM POS tagging task.

We propose a *first-of-its-kind* work in generating CM annotated data. We introduce the PACMAN dataset, *generated* using an *alignment*, *annotation* and *replacement* strategy from *Hindi-English* parallel corpus. We claim that PACMAN is the largest CM annotated dataset (around *50K* samples). Although the generated dataset is for *Hinglish*, the strategy can be transferred to any CM language pair, having available *parallel corpus* and *POS taggers*.

The PACMAN data adheres to standard spelling and grammar, unlike social media data, primarily used in CM research work. The use of both *Hindi* and *English* as *matrix languages*, ensures uniform distribution of words in both languages, and the potential to understand monolingual contexts.

To establish the effectiveness of the dataset, we build both ML (*CRF*) and DL (*BiLSTM*) based sequential labeling models on PACMAN data. Unlike previous work, our DL model outperforms the ML model by 1.5%.

We analyse the effect of *word-level language information* on the CM POS tagging task, which reveals that, with a larger dataset, cases where WLI is crucial, are minuscule, thus elevating the need for WLI in CM POS labeling.

We validate our dataset against the existing benchmark dataset for CM POS tagging. Our best model outperforms the SOTA benchmark by 10% and in all computational scenarios, signifying that our dataset is more generalised and capable of handling a wide spectrum of CM data as well as monolingual data.

Scrutiny of errors observed in our dataset manifests that inaccuracies in probabilistic tools and resources used as a part of the data generation pipeline, adversely affect the quality of the dataset, although such cases are scarce.

In future, we endeavour to generate more data in order to build *transformer-based* models on PACMAN, and use *matrix language* information to prevent errors actuated by structural differences between *Hindi* and *English*.

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.

Peter Auer. 2020. *The pragmatics of code-switching: a sequential approach*, pages 123–138.

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.

Stephen Boggs. 1983. Discourse strategies . john j. gumperz. *American Anthropologist*, 85.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Arindam Chatterjee, Vineeth Guptha, Parul Chopra, and Amitava Das. 2020. Minority positive sampling for switching points - an anecdote for the code-mixing language modeling. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 6228–6236, Marseille, France. European Language Resources Association.

Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Identifying sources of opinions with conditional random fields and extraction patterns. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 355–362, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *NAACL*.

Björn Gambäck. 2014. On measuring the complexity of code-mixing.

Shalini Ghosh, Oriol Vinyals, Brian Strope, Scott Roy, Tom Dean, and Larry P. Heck. 2016. Contextual LSTM (CLSTM) models for large scale NLP tasks. *CoRR*, abs/1602.06291.

J.J. Gumperz. 1964. Hindi-punjabi code-switching in delhi. *Proceedings of the Ninth International Congress of Linguists*, pages 1115–1124.

Anupam Jamatia, Björn Gambäck, and Amitava Das. 2015. Part-of-speech tagging for code-mixed english-hindi twitter and facebook chat messages. In *RANLP*.

Navya Jose, Bharathi Raja Chakravarthi, Shardul Suryawanshi, Elizabeth Sherly, and John P. McCrae. 2020. A survey of current datasets for code-switching research. *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 136–141.

Aravind K. Joshi. 1982. Processing of sentences with intra-sentential code-switching. In *Proceedings of the 9th Conference on Computational Linguistics - Volume 1*, COLING '82, page 145–150, CZE. Academia Praha.

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 Language Resources Association (ELRA).

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.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *MATHEMATICAL PROGRAMMING*, 45:503–528.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Carol Myers-Scotton. 1993. Duelling languages: grammatical structure in code-switching.

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

Fuchun Peng and Andrew McCallum. 2006. Information extraction from research papers using conditional random fields. *Information Processing Management*, 42(4):963–979.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2089–2096, Istanbul, Turkey. European Language Resources Association (ELRA).

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. *CoRR*, abs/2003.07082.

Vincenz Schwab. 2021. *Codemixing und Codeswitching: Volkssprachige Inserte in Rechtsquellen des Frühmittelalters*, pages 75–90.

Kushagra Singh, Indira Sen, and Ponnurangam Kumaraguru. 2018. A twitter corpus for hindi-english code mixed pos tagging. pages 12–17.

Thamar Solorio and Yang Liu. 2008. Part-of-speech tagging for english-spanish code-switched text. pages 1051–1060.

Vivek Srivastava and Mayank Singh. 2021. Hinge: A dataset for generation and evaluation of code-mixed hinglish text. *CoRR*, abs/2107.03760.

Mark Thompson. 2009. Brenda danet susan c. herring (eds.), the multilingual internet: Language, culture and communication online. *Language in Society - LANG SOC*, 38.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, page 173–180, USA. Association for Computational Linguistics.

Yogarshi Vyas, Spandana Gella, Jatin Sharma, Kalika Bali, and Monojit Choudhury. 2014. Pos tagging of english-hindi code-mixed social media content. pages 974–979.

Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao. 2015. A unified tagging solution: Bidirectional LSTM recurrent neural network with word embedding. *CoRR*, abs/1511.00215.

A Appendix

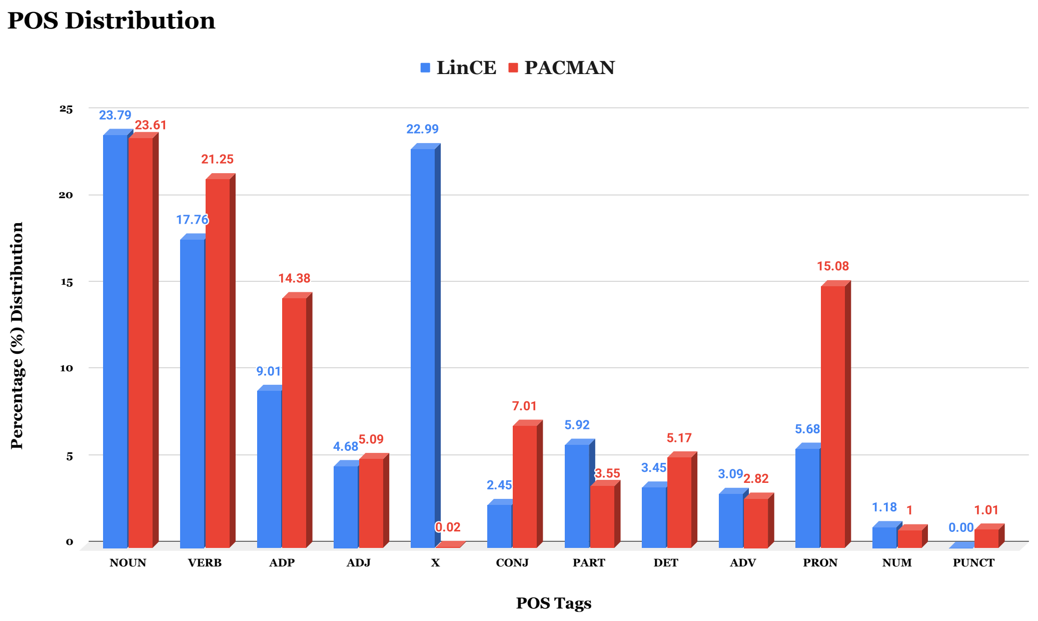
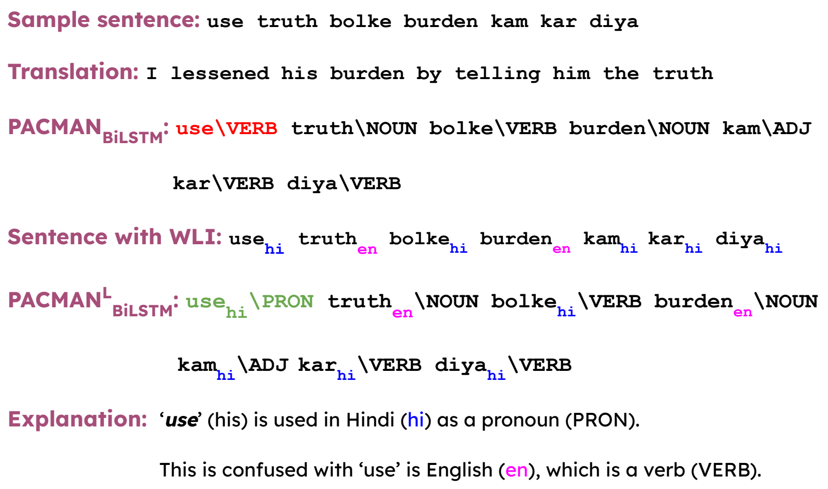


Figure 4: Percentage-wise comparison of POS distribution between PACMAN and LinCE over 12 universal POS tags. The high percentage of the 'X' POS tag indicates that the LinCE data is social media based and hence noisy.



Sample sentence: use truth bolke burden kam kar diya

Translation: I lessened his burden by telling him the truth

PACMAN$_{BiLSTM}$: use\VERB truth\NOUN bolke\VERB burden\NOUN kam\ADJ kar\VERB diya\VERB

Sentence with WLI: use$_{hi}$ truth$_{en}$ bolke$_{hi}$ burden$_{en}$ kam$_{hi}$ kar$_{hi}$ diya$_{hi}$

PACMAN$^{L}_{BiLSTM}$: use$_{hi}$\PRON truth$_{en}$\NOUN bolke$_{hi}$\VERB burden$_{en}$\NOUN kam$_{hi}$\ADJ kar$_{hi}$\VERB diya$_{hi}$\VERB

Explanation: '***use***' (his) is used in Hindi (hi) as a pronoun (PRON). This is confused with 'use' is English (en), which is a verb (VERB).

Figure 5: Example showing where word-level language information (WLI) helps in correct POS identification. It can be seen that the word '*use*' (Hindi for *his*) has different meanings but the same spelling in English and transliterated Hindi. With WLI, this confusion is resolved, and the correct POS tag is predicted.

Error Corpora for Different Informant Groups: Annotating and Analyzing Texts from L2 Speakers, People with Dyslexia and Children

Þórunn Arnardóttir, Isidora Glišić, Annika Simonsen,
Lilja Björk Stefánsdóttir, Anton Karl Ingason
University of Iceland
Sæmundargata 2, 102 Reykjavík, Iceland
{thar, isg14, ans72, lbs, antoni}@hi.is

Abstract

Error corpora are useful for many tasks, in particular for developing spell and grammar checking software and teaching material and tools. We present and compare three specialized Icelandic error corpora; the Icelandic L2 Error Corpus, the Icelandic Dyslexia Error Corpus, and the Icelandic Child Language Error Corpus. Each corpus contains texts written by speakers of a particular group; L2 speakers of Icelandic, people with dyslexia, and children aged 10 to 15. The corpora shed light on errors made by these groups and their frequencies, and all errors are manually labeled according to an annotation scheme. The corpora vary in size, consisting of errors ranging from 7,817 to 24,948, and are published under a CC BY 4.0 license. In this paper, we describe the corpora and their annotation scheme, and draw comparisons between their errors and their frequencies.

1 Introduction

Error analysis is a crucial part of corpus linguistics and applied linguistics as it provides an insight into language use and the needs of speaker groups within a language. It thereby facilitates the development of a variety of practical tools to aid these needs, such as more focused teaching and learning materials, and software tools like spelling and grammar checkers. To contribute to this field, we present three Icelandic error corpora, each focusing on different speaker populations; second language users of Icelandic (hereinafter: L2 Icelandic), children at the age of 10 to 15, and people with dyslexia. Thus, we have created three manually annotated error corpora of different sizes, one for each respective informant group, and extracted statistical data on the errors that occur. These corpora are an invaluable source for further research in Icelandic for both academic and practical purposes. All corpora are published under a CC BY 4.0 license. (Ingason et al., 2022b,a, 2021)

The paper is structured as follows. Section 2 discusses the creation of error corpora in general. Section 3 describes the specialized error corpora, their annotation and the three respective corpora. Section 4 draws comparisons between the errors in the three specialized corpora and compares them to errors in a previous general error corpus. Section 5 discusses possible future use of the corpora and finally, we conclude with Section 6.

2 Creating Error Corpora

Error analysis has been an integrated part of applied linguistics and computational linguistics for decades, and corpus linguistics in general has developed as a key methodology in the humanities and social sciences (Paquot and Gries, 2020). It provides key insight into both the errors that adult native speakers of a language produce in writing, as well as those of language learners, children, and people with different learning difficulties such as dyslexia. Gathering data on these errors has become a standard practice for many languages and is invaluable for creating different software tools for language correction and suggestion, such as spell checkers, grammar assistance, and lexical and stylistic suggestions. Furthermore, an error corpus gives way to contrastive analysis which leads to better understanding of language use in different groups and the creation of both digital and analogue content that would facilitate them (e.g. improving teaching materials for second language learners and children).

The Icelandic Error Corpus was created for this purpose, and was the first Icelandic error corpus (Arnardóttir et al., 2021). It has already been used for developing an Icelandic open-source spell and grammar checker (Óladóttir et al., 2022), wherein the labeled errors in the corpus are used to measure the spell and grammar checker's improvements. This error corpus consists of texts written by Icelandic native-speaking adults with no known learning disabilities, and provides information on errors which this speaker group is likely to make. However, the Icelandic population consists of various speakers who might make errors different to a general speaker. For this purpose, error corpora for particular speaker groups are important. Analyzing errors made by these groups enables the development of spell and grammar checkers and practical tools suited for those groups, as well as facilitating effective teaching methods and materials.

3 The Icelandic Specialized Error Corpora

Three Icelandic error corpora were created between the fall of 2019 and the fall of 2022, reflecting three different user groups; The Icelandic L2 Error Corpus, The Icelandic Dyslexia Error Corpus, and The Icelandic Child Language Error Corpus. All corpora are published under a CC BY 4.0 license in the Icelandic CLARIN repository (Ingason et al., 2022b,a, 2021). An older version of the Icelandic L2 Error Corpus was described in Glisic and Ingason (2022).

3.1 Annotation

The Icelandic Specialized Error Corpora all have the same annotation scheme and structure, which is shared by the Icelandic Error Corpus (Arnardóttir et al., 2021). The steps involved in creating the corpora were gathering large quantities of texts within each focus category, manually proofreading the texts for errors, and finally creating the corpus in the decided digital format. Each error in the texts was then manually labeled within a pre-decided annotation scheme. The corpora are published in augmented TEI-format XML documents, making them machine readable so that corpus management platforms particular to TEI format files can be used to obtain information from the corpus. A specific TEI element, *revision*, was created to map out the differences between the original text file and the manually corrected file. Each XML document consists of many revision spans that include the mismatching text and one or more error tags that are manually classified within a previously decided annotation scheme.

Errors in the original texts were detected by following Icelandic spelling and grammar rules. Many of these rules are included in the Icelandic language council's spelling rules.[1] Rules on language usage are included in a resource called *Málfarsbankinn*[2] (direct translation: *The Language Usage Bank*). This is a collection of rules and general advice concerning grammar, fixed phrases, spelling, and more. In addition to these explicit errors, stylistic errors were also corrected, i.e. errors which are not included in the aforementioned resources, but belong to known guidelines for writing text. These errors for example include using numerals instead of numbers in particular cases.

Language error classification can be done in many different ways, but two major categories are mostly defined as linguistic errors (morphology, syntax, etc.) and surface structure taxonomies (omission, addition, etc.), where most studies combine the analysis of both these categories (Macdonald et al., 2013). This practice was adopted for the Icelandic error corpora and a particular annotation scheme was created. It evolved as the error annotation progressed and new types of texts came in — particularly many new error categories were noted with L2 texts (more on this in Subsection 3.2).

The annotation scheme is hierarchical with three layers. Errors are classified within five main categories: orthography, grammar, vocabulary, coherence, and style. Each main category is further divided into more descriptive subcategories, which are then divided into error codes, 258 in total.[3] Each error code describes a specific type of error, although the scope and particularity can vary. For example, the code 'af4að' is used when the preposition *að* 'to' is mistakenly replaced by *af* 'of', whereas 'wrong-prep' is used in general with incorrect prepositions. 'i4y' is used when letters "i" and "y" are mixed up in a word but 'letter-rep' is used when a letter incorrectly replaces another one. This difference in scope is because both initial analysis and previous research on e.g. learner language indicated that certain specific errors occurred quite frequently, and the more detailed the annotation system, the better insight we can have into these errors, which will be of great value for future research. 'Wording' is the most general error type, and includes any type of formulating a phrase or a clause in a wrong way. Finally, some detected errors are connected to another error(s) within the sentence, such as in 'wording' and errors connected

[1]https://ritreglur.arnastofnun.is

[2]http://malfar.arnastofnun.is

[3]The annotation scheme is accessible at https://github.com/antonkarl/iceErrorCorpusSpecialized/blob/master/errorCodes.tsv.

to syntax. These are classified within a separate category, 'other', which includes only one error code, 'dep', representing a dependent error.

Initial work on the Icelandic error corpus started in the autumn of 2019, and as of January 2020, several proofreaders were working with the texts; at one point a total of 12 people were reading over and correcting. Five specialists (either language technologists or Icelandic language specialists) worked on converting the texts into corpus data, creating the annotation system, and finally categorizing the errors found in the texts. Texts in the specialized error corpora needed to be collected from private sources, since no freely accessible texts written by these user groups were available. This proved to be a difficult and time-consuming process, because awareness about the project had to be raised within the interest groups and they had to be encouraged to participate. Interested authors signed publication agreements, which differed between user groups, and is discussed further in the sections pertaining to each corpus.

3.2 The Icelandic L2 Error Corpus

Icelandic is an increasingly popular language among language learners; there is a sizable population of immigrants in Iceland, who learn Icelandic to integrate into society. Additionally, there are people who are interested in learning Icelandic because they are language enthusiasts. However, teaching materials for Icelandic as a second language are scarce and in high demand. The creation of an L2 error corpus is a major step towards facilitating better teaching materials and also language learning tools (Glisic and Ingason, 2022). The version of the Icelandic L2 Error Corpus which is discussed here is an improved version of the one discussed in Glisic and Ingason (2022). More data has been added, and as a result, more errors have been collected.

The Icelandic L2 Error Corpus is a collection of 101 texts, predominantly student essays, written by 44 non-native speakers of Icelandic with 17 different native languages, containing in total 24,948 error instances in 17,241 revisions. Table 1 shows this, along with word count and frequency of errors per 1,000 words, which is 153.93.

Figure 1 and Table 2 depict the error rate per 1,000 words based on skill level, where the width of the bars indicates the number of words submitted for each level. Skill levels are shown according to the Common European Framework of Reference for Languages (CEFR), which is an international standard for describing language ability. It describes language ability on a six-point proficiency scale – A1, A2, B1, B2, C1, C2. 'A' is considered the beginner level, 'B1' intermediate, 'B2' advanced and 'C' proficient (near-native) level (North and Piccardo, 2020). As mentioned, the corpus as a whole has 153.93 errors per 1,000 words, but the number of errors varies based on the authors' accomplished skill level and steadily drops in accordance with the language learning progress.

Revisions	Errors	Files	Words	Errors/1,000w
17,241	24,948	101	162,071	153.93

Table 1: Number of revisions, errors, files, words, and errors per 1,000 words in the Icelandic L2 Error Corpus

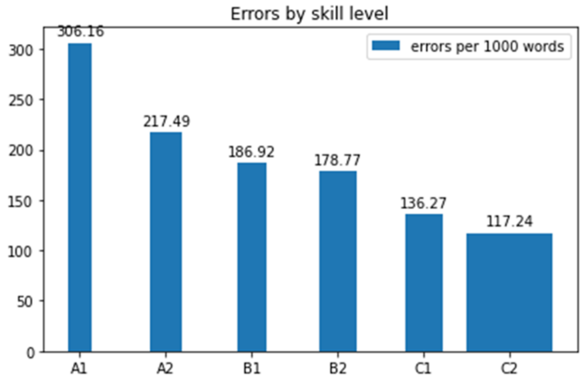


Figure 1: Error rate per 1,000 words in the Icelandic L2 Error Corpus according to learner level.

Level	Files	Total words	Total errors	Errors/1,000w
A1	20	7,960	2,437	306.16
A2	22	14,695	3,196	217.49
B1	14	16,071	3,196	186.92
B2	14	21,447	3,834	178.77
C1	16	27,871	3,798	136.27
C2	15	74,027	8,679	117.24

Table 2: Number of files, words, errors, and errors per 1,000 words in the Icelandic L2 Error corpus according to proficiency level.

In Table 3, we display the top 10 most common errors in the Icelandic L2 Error Corpus. The most common error is 'wording', which makes up 10.96% of the errors. 'Punctuation' and 'inflection' follow closely behind at 9.60% and 9.04%. Bear in mind that here we include L2 speakers of all proficiency levels. It is interesting to note that while 'inflection' is the third most common error in the L2 corpus, it is not even in the top 10 most common errors in the Icelandic Error Corpus (Arnardóttir et al., 2021).

Subcategory	Main category	n	prop
wording	style	2735	11.0
punctuation	orthography	2396	9.6
inflection	grammar	2256	9.0
miscellaneous	other	1895	7.6
agreement	grammar	1526	6.1
prep	grammar	1452	5.8
definitiveness	grammar	1186	4.8
typo	orthography	1153	4.6
syntax	grammar	1146	4.6
insertion	vocabulary	1133	4.5

Table 3: Top 10 most frequent errors in the Icelandic L2 Error Corpus according to subcategory.

As mentioned in Subsection 3.1, the texts were previously unpublished and obtained directly from their authors. The text collection effort lasted from September 2020 to May 2022. The call for texts was first directed to the students of Icelandic as a second language at the University of Iceland, but was subsequently extended to a public call. As a result of the call to students at the university, the texts are mainly student essays submitted for evaluation in various courses at the University of Iceland. Authors signed publication agreements, wherein they stated their native language, agreed to the text being published, and could choose to be anonymous or not. Out of 44 authors, seven chose to not be anonymous.

3.3 The Icelandic Dyslexia Error Corpus

Dyslexia is a learning disability, causing difficulty with reading but also with writing. In general, people with dyslexia make more misspellings than people who do not have it, and also different types of errors, which can affect how useful general spell and grammar checkers are for people with dyslexia. This difference in error rate and error type has not been previously studied for Icelandic informants, and for that reason, the Icelandic Dyslexia Error Corpus was created.

The dyslexia texts were collected through an open call. The only criteria for the texts was that their authors' native language was Icelandic, and that they had been diagnosed with dyslexia, but no proof of a diagnosis was asked for. The texts were written by informants born between 1961 and 2004 and some texts were written by the same author. For ethical reasons, as dyslexia is a medically diagnosed disorder, no more information on the authors was retained.

The Icelandic Dyslexia Error Corpus consists of 35 files totaling 38,891 words, and has 5,075 revisions with 8,436 errors (see Table 4). The error rate in the Icelandic Dyslexia Error Corpus is 216.91 errors per 1,000 words, which is the highest error rate of all the corpora, exceeded only by the A1- and A2-level L2 speakers.

Revisions	**Errors**	**Files**	**Words**	**Errors/1,000w**
5,075	8,436	35	38,891	216.19

Table 4: Number of revisions, errors, files, words, and errors per 1,000 words in the General Dyslexia Error Corpus.

Table 5 displays the 10 most frequent errors in the Icelandic Dyslexia Error Corpus, according to subcategory. The corpus shares similarities with the Icelandic Error Corpus in that among the most common errors are 'punctuation' and 'wording', but unlike the other corpora, the dyslexia corpus has a higher proportion of typos; a characteristic which is expected of people with dyslexia. The 'nonword' error code (a non-compound that does not appear in the dictionary) is also relatively high compared to the other corpora.

Subcategory	Main category	n	prop
typo	orthography	905	10.7
punctuation	orthography	903	10.7
wording	style	812	9.6
nonword	orthography	758	9.0
miscellaneous	other	545	6.5
syntax	grammar	524	6.2
spacing	orthography	443	5.2
insertion	vocabulary	411	4.9
omission	vocabulary	361	4.3
spelling	orthography	355	4.2

Table 5: Top 10 most frequent errors in the Icelandic Dyslexia Error Corpus according to subcategory.

Texts included in this corpus were collected over a two-year period, between October 2020 and October 2022, by different means. A collaboration with the Icelandic Dyslexia Association and school counselors at Icelandic colleges was established, and an open call to people with dyslexia was sent out. Authors who submitted their texts signed a publication agreement, wherein they confirmed that they had been diagnosed with dyslexia and agreed that their texts would be published. Additionally, they could choose to be anonymous or not, but only two authors chose not to be anonymous.

3.4 The Icelandic Child Language Error Corpus

The Icelandic Child Language Error Corpus consists of 119 texts written by children aged 10 to 15 (born between years 2005 and 2010). The corpus excludes texts which were written by children with dyslexia as well as any text written by a child whose first language is not Icelandic. The interest in children's texts arose from the need to gain plausible insight into their vocabulary and grammar use, and the struggles they face in the process of language acquisition and learning to write. So far, we have been aware that children do not make the same mistakes as adults in their writing, based on tentative assumptions, teachers' experience and some studies of child language from other languages. Creating this corpus provides a unique opportunity to map out the exact errors and apply the findings directly in facilitating language and literacy development on elementary school level.

The corpus contains 7,817 errors, with an error rate of 208.77 per 1,000 words, as seen in Table 6. This is the second highest overall error rate in the four corpora, after the dyslexia corpus, although the L2 informants at proficiency level A1 and A2 have the highest error rates.

Revisions	**Errors**	**Files**	**Words**	**Errors/1,000w**
5,079	7,817	119	37,443	208.77

Table 6: Number of revisions, errors, files, words, and errors per 1,000 words in the Icelandic Child Language Error Corpus.

Table 7 shows the top 10 most common errors in The Icelandic Child Language Error Corpus. Similar to the general corpus and the dyslexia corpus, the most common error in the child language corpus is 'punctuation', with 18.92% frequency, and the second most common error is 'wording', which comprises 10.63% of the errors in the corpus. Here we also see some predictable children's mistakes such as regarding capitalization, which is not common in the Icelandic Error Corpus, the L2 Error Corpus or the Dyslexia Error Corpus, which were written by adults.

The text collection effort lasted from February 2021 to September the same year. In order to collect the published texts, two methods were chosen; first, an open call was made to any parents with children of the appropriate age. This resulted in a few texts, but most of them were collected by means of collaborations with Icelandic elementary schools. Written assignments were collected with the help of teachers and the children's guardians signed publication agreements for publication of the texts. All authors are anonymous, have not been diagnosed with dyslexia and Icelandic is their native language.

Subcategory	Main category	Frequency (%)
punctuation	orthography	18.9
wording	style	10.6
miscellaneous	other	8.0
capitalization	orthography	6.7
insertion	vocabulary	6.1
nonword	orthography	5.9
typo	orthography	4.7
omission	vocabulary	4.6
syntax	grammar	4.3
spacing	orthography	4.2

Table 7: Top 10 most frequent errors in the Icelandic Child Language Error Corpus according to subcategory.

4 Error Analysis

The error corpora for Icelandic are of varying size (see Table 8). The proportion of L2 corpus text reflects the population of Iceland (around 15–20% of the population are immigrants), but the data from children does not reflect population numbers. Comparing error frequencies between the corpora, we can infer that people with dyslexia make the most errors and children the second most errors. However, as has been discussed previously, L2 speakers at proficiency levels A1 and A2 make more errors than both speaker groups, with 306.16 and 217.49 errors per 1,000 words, respectively.

Table 9 shows the 5 most common errors, according to subcategory, in each of the three specialized error corpora and the Icelandic Error Corpus[4] (titled 'General' here), along with each error's proportion. We see that the different error corpora share many of the most common errors, e.g. 'punctuation' and 'wording'. Note that the 'miscellaneous' subcategory only consists of dependent errors, which are often connected to errors relating

[4]This information is taken from Arnardóttir et al. (2021).

Corpus	Number of words	Number of errors	Errors per 1,000/w
L2	162,071	24,948	153.93
Dyslexia	38,891	8,436	216.91
Children	37,443	7,817	208.77

Table 8: Number of words, errors and errors per 1,000 words in the three error corpora.

General	%	L2	%	Dyslexia	%	Children	%
punctuation	25.46	wording	11.00	typo	10.7	punctuation	18.9
wording	14.74	punctuation	9.6	punctuation	10.7	wording	10.6
spacing	6.98	inflection	9.0	wording	9.6	miscellaneous	8.0
nonword	6.11	miscellaneous	7.6	nonword	9.0	capitalization	6.7
typo	5.68	agreement	6.1	miscellaneous	6.5	insertion	6.1

Table 9: Frequency of the 5 most common errors in the error corpora according to subcategory.

to wording.

The L2 error corpus is the only corpus with grammatical errors as the most common ones, i.e. 'inflection' and 'agreement', which reflects the fact that the authors' native language is not Icelandic. Inflectional errors include errors where a word is in the wrong case, e.g. when a subject of a sentence should be in the nominative case but is instead in the dative case. Agreement errors include e.g. when a finite verb is not in agreement with a noun phrase, e.g. when it comes to number.

As mentioned in Section 3.3, among the most common errors in the dyslexia corpus is 'typo', which is in accordance with what is to be expected of dyslexic writers. This error is two times more frequent in the dyslexia corpus as compared to the general corpus, and is not among the 5 most common errors in children's text. Typos are e.g. errors where a letter within a word is incorrectly replaced by a different letter, or a letter is missing within a word.

As mentioned in Section 3.4, the children's corpus shows more frequent capitalization errors than in the other corpora, but it is also the only corpus to have an error relating to vocabulary, 'insertion', among the 5 most common errors. Insertion errors are e.g. errors where a redundant conjunction or word appears.

Certain error codes only occur in certain corpora. This is illustrated in Table 10. The dyslexia corpus does not contain any unique error codes, but the L2 and children's corpora do. It is therefore possible to see if there is any specific type of error that only a certain speaker is more likely to make. Most error codes pertain to punctuation, but two errors in the L2 corpus, 'adj4noun' and 'þar4það', are both lexical. The former is when an adjective incorrectly replaces a noun and the latter is when the word *þar* 'there' is written instead of *það* 'it'.

L2	Children
adj4noun	ex4qm
þar4það	
semicolon4conjunction	
wrong-symbol	

Table 10: Error codes that only appear in certain corpora.

The error codes in each corpus were ranked by frequency of occurrence (starting with 1 for the most commonly occurring error and total number of error codes that appear in the corpus, plus 1 for the ones that never appear in it) and then compared between the corpora, using the general Icelandic Error Corpus as the default. Ranking comparison produced a *delta rank*, which is the difference between the frequency rank of a certain error between corpora, and this was extracted in Tables 11, 12, and 13, which show the 10 highest delta ranks when compared to the general corpus. This clearly shows that some error codes pertaining to grammar and lexical issues are much more frequent in the specialized corpora than in the general corpus, but interestingly enough, the delta rank is similar for each of the specialized corpora.

5 Future Use

The error corpora for Icelandic can be used separately for specific use cases, but they can also be merged for a general overview of the different types of speakers that exist in the population.

Error code	Rank General	Rank L2	Delta rank
context	132	5	127
syntax-other	132	13	113
missing-hyphen	10	104	94
v3	131	41	90
extra-sub	126	45	81
extra-prep	97	18	79
genitive	102	34	68
tense4perfect	126	63	63
extra-hyphen	46	108	62
extra-dem-pro	131	69	62

Table 11: Error codes with the highest delta rank between the general Icelandic Error corpus and the L2 corpus.

Error code	Rank General	Rank Dyslexia	Delta rank
context	132	31	101
syntax-other	132	44	88
extra-fin-verb	126	40	86
extra-sub	126	48	78
hyphen4endash	115	39	76
extra-prep	97	27	70
new-passive	124	56	68
extra-inf-part	129	63	66
v3	131	67	64
extra-dem-pro	131	69	62

Table 12: Error codes with the highest delta rank between the general Icelandic Error corpus and the Dyslexia corpus.

With the general overview, we can see how an L2 learner is going to make different errors than a native speaker, while a child will make different errors than an adult, etc. The current version of the L2 error corpus consists of texts written by adults who are only learning Icelandic. By collecting texts from learners who are learning more than one language, it would be possible to determine whether the types of errors that learners make are similar across languages, which may help in pooling larger data sources and transfer learning across languages.

Furthermore, combining all the specialized error corpora can facilitate a spellchecker that takes into account the needs of all these varieties of speakers, and can therefore detect and correct errors which are often produced by them. The error corpora can be put into practical use in creating a grammar and spelling correction software, in the same way as the Icelandic Error Corpus has been used (Óladóttir et al., 2022). Furthermore, experiments on using them as fine-tuning data for a neural spell and grammar checker have already begun (Ingólfsdóttir et al., 2022).

An error corpus can also be used to create other tools, such as language learning tools and teaching materials. Statistics and error examples from the Icelandic L2 Error Corpus can be used for developing computer-assisted language learning tools, such as flashcards (Xu and Ingason, 2021). An L2 error corpus also sheds light on learner interlanguage, which provides insight into how grammatical and lexical categories are acquired and internalized (Glisic and Ingason, 2022). This insight can be used when developing language learning tools for Icelandic, and it can also be helpful for teachers who teach Icelandic as a second language, because it helps them predict which errors the language learners will make at what stage in their proficiency level. This is also the case for teachers who teach Icelandic students with dyslexia; the Icelandic Dyslexia Error Corpus documents the most common errors made by the speakers – a valuable insight into what dyslexia looks like in Icelandic. Furthermore, the Icelandic Child Language Error Corpus can be used when teaching children how to write.

Error code	Rank General	Rank Children	Delta rank
extra-sub	126	18	108
context	132	43	89
extra-fin-verb	126	41	85
lower4upper-initial	89	8	81
extra-dem-pro	131	55	76
missing-qm	117	43	74
v3	131	59	72
syntax-other	132	61	71
extra-inf-part	129	60	69
new-passive	124	57	67

Table 13: Error codes with the highest delta rank between the general Icelandic Error corpus and the Child language corpus.

6 Conclusion

We have described three new Icelandic error corpora: the Icelandic L2 Error Corpus (Ingason et al., 2022b), the Icelandic Dyslexia Error Corpus (Ingason et al., 2022a) and the Icelandic Child Language Error Corpus (Ingason et al., 2021). All corpora are published under a CC BY 4.0 license and reflect errors made by the three user groups. The value of such corpora is diverse; they can be used to develop user-oriented spell and grammar checkers, can guide the development of language learning tools and can help in teaching Icelandic to non-native speakers, and in teaching dyslexic students or children in general to write.

Acknowledgements

This project was funded by the Language Technology Programme for Icelandic 2019–2023. The programme, which is managed and coordinated by Almannarómur (https://almannaromur.is/), is funded by the Icelandic Ministry of Education, Science and Culture. We would like to thank the anonymous reviewers for their valuable feedback.

References

Þórunn Arnardóttir, Xindan Xu, Dagbjört Guðmundsdóttir, Lilja Björk Stefánsdóttir, and Anton Karl Ingason. 2021. Creating an error corpus: Annotation and applicability. In *Proceedings of CLARIN 2021*, pages 59–63.

Isidora Glisic and Anton Karl Ingason. 2022. The nature of Icelandic as a second language: An insight from the learner error corpus for Icelandic. *Linköping Electronic Conference Proceedings*.

Anton Karl Ingason, Þórunn Arnardóttir, Lilja Björk Stefánsdóttir, and Xindan Xu. 2021. The Icelandic child language error corpus (IceCLEC) version 1.1. CLARIN-IS.

Anton Karl Ingason, Þórunn Arnardóttir, Lilja Björk Stefánsdóttir, Xindan Xu, Dagbjört Guðmundsdóttir, and Isidora Glišić. 2022a. The Icelandic dyslexia error corpus 1.2 (22.10). CLARIN-IS.

Anton Karl Ingason, Lilja Björk Stefánsdóttir, Þórunn Arnardóttir, Xindan Xu, Isidora Glišić, and Dagbjört Guðmundsdóttir. 2022b. The Icelandic L2 error corpus (IceL2EC) 1.3 (22.10). CLARIN-IS.

Svanhvít Lilja Ingólfsdóttir, Pétur Orri Ragnarsson, Haukur Páll Jónsson, Haukur Barri Símonarson, Vilhjálmur Þorsteinsson, and Vésteinn Snæbjarnarson. 2022. Byte-level neural error correction model for Icelandic - yfirlestur (22.09). CLARIN-IS.

Penny Macdonald, Amparo García-Carbonell, and Sierra Jose Miguel Carot. 2013. Computer learner corpora: Analysing interlanguage errors in synchronous and asynchronous communication. *Language Learning & Technology*, 17:36–56.

Brian North and Enrica Piccardo. 2020. *Common European Framework of Reference for Languages: Learning, Teaching, Assessment. Companion volume with new Descriptors*.

Hulda Óladóttir, Þórunn Arnardóttir, Anton Ingason, and Vilhjálmur Þorsteinsson. 2022. Developing a spell and grammar checker for Icelandic using an error corpus. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 4644–4653, Marseille, France. European Language Resources Association.

Magali Paquot and Stefan Th. Gries. 2020. *A Practical Handbook of Corpus Linguistics*. Springer Cham, Switzerland.

Xindan Xu and Anton Karl Ingason. 2021. Developing Flashcards for learning Icelandic. In *Proceedings of the 10th Workshop on NLP for Computer Assisted Language Learning*, pages 55–61, Online. LiU Electronic Press.

Similarity Based Label Smoothing For Dialogue Generation

Sougata Saha, Souvik Das, Rohini Srihari
State University of New York at Buffalo
Department of Computer Science and Engineering
`{sougatas, souvikda, rohini}@buffalo.edu`

Abstract

Generative neural conversational systems are typically trained by minimizing the entropy loss between the training "hard" targets and the predicted logits. Performance gains and improved generalization are often achieved by employing regularization techniques like label smoothing, which converts the training "hard" targets to soft targets. However, label smoothing enforces a data-independent uniform distribution on the incorrect training targets, leading to a false assumption of equiprobability. In this paper, we propose and experiment with incorporating data-dependent word similarity-based weighing methods to transform the uniform distribution of the incorrect target probabilities in label smoothing to a more realistic distribution based on semantics. We introduce hyperparameters to control the incorrect target distribution and report significant performance gains over networks trained using standard label smoothing-based loss on two standard open-domain dialogue corpora.

1 Introduction

Response generators rely heavily on language modelling for response generation. Given a context comprising multiple conversation utterances, a response generator is formulated as a next utterance prediction problem, where the task is to generate a response conditioned on the context. With the advent of deep learning and availability of sufficient training data, parametric models like recurrent neural networks and transformers are generally used for language modelling. Trained by minimizing the expected cross entropy between the training targets and the prediction logits, such models often overfit the training data and does not generalize well on the test set. Label smoothing proposed by Szegedy et al. (2015) to improve the performance of Inception net image classifier on the ImageNet dataset has gained wide acceptance in Natural Language Processing tasks as a regularization tech-

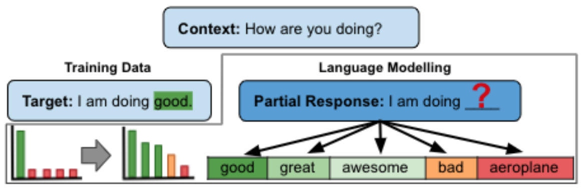


Figure 1: Sample conversation depicting token probability distribution using label smoothing, in comparison to desired distribution.

nique to enhance the generalization capability of deep neural networks. Vaswani et al. (2017) in his work "Attention is all you need", where he proposed the state-of-the-art transformer architecture, had reported performance gains in machine translation using label smoothing during training. Unlike other regularization techniques which constrain the model parameters and hidden representations, label smoothing augments the actual targets by reducing the target probability and assigning low probabilities to all classes, following a data independent uniform distribution. Thus, preventing the model from predicting the correct labels overconfidently during training. However, as pointed out by Pereyra et al. (2017) and Hinton et al. (2015), the probabilities assigned to both the correct and incorrect classes constitute the knowledge of a network. In language modelling, incorporating label smoothing and assigning a uniform probability to all the incorrect classes can convey a false knowledge to the model. For example, as depicted in Figure 1, while generating "I am doing good" in response to the query "How are you doing ?", having generated the partial phrase "I am doing", both "great" and "awesome" conveys the same message as "good". Although "bad" would convey an opposite yet meaningful message, a random word like "aeroplane" would be inappropriate. Hence, instead of assuming a uniform distribution for the incorrect classes while using label smoothing, we can incorporate a weighing mechanism to present such knowledge to the model. Here, we introduce simple ways of imparting such information by modifying the data independent uniform distribution in label smoothing with a more appropriate data dependent distribution proportional to the pre-trained word-embedding based similarity between the actual and incorrect targets. Our primary contributions are follows (i) We propose a robust mechanism for augmenting the target labels in language modelling, which better reflects the real-world. (ii) We experiment our proposed framework with different hyper-parameter settings and perform thorough analysis of the observations [1]

2 Related Work

Although numerous techniques have been introduced to enhance the generalizability of neural networks, as pointed out by Pereyra et al. (2017), most work focus on regularizing model parameters, compared to external regularization techniques like label smoothing or target data augmentation. Recent approaches for generalizable conversations can be broadly categorized as follows

Loss function augmentation: Li et al. (2016) proposed using Maximum Mutual Information along with the Cross Entropy loss, in order to penalize generic responses like "I do not know", which are frequent in conversational datasets. Jiang et al. (2019) attributed generic responses to the cross entropy loss function, which prefers frequent tokens. They proposed augmenting the loss function with a frequency based weighing mechanism dependent on the corpus for engendering diverse responses. Wang et al. (2020) experimented with using optimal transport to match sequences generated in the teacher and student modes, and increasing performance of student forced networks on the test dataset by reducing the gap between the two modes. Wang et al. (2021) proposed an adaptive label smoothing approach that can adaptively estimate a target label distribution at each time step for different contexts. Compared to their approach, our proposed method is simpler with fewer parameters.

Data augmentation: Cai et al. (2020) demonstrated that conversational datasets generally don't exhibit coherence in query response pairs, which affect the Cross Entropy loss. They propose a training data augmentation module, which can not only replace words in the actual target response with similar words using BERT (Devlin et al., 2019), but also augment the style of the response, preserving the meaning. They further introduced a neural weighting mechanism, which can assign weights or importance to the augmented and golden training data, and report significant performance gains. Kang and Hashimoto (2020) demonstrated that the log loss is not robust to noise, and hence proposed truncating the distribution of the training targets to achieve an easy to optimize and more robust loss function. He and Glass (2020) introduced a network which can provide negative generated samples, and train the generation model to maximize the log likelihood of training data while minimizing the likelihood of negative samples. Since instead of augmenting the training data, we adjust the probability of incorrect labels for each correct label, our proposed method belongs to the first category.

3 Methods and Experiments

We experiment with ways to augment the data independent uniform distribution enforced by label smoothing. Let U_i be an utterance consisting of words $\{w_j\}_{j=1}^{N}$, where N is the number of words in the utterance. For each word w_j, in label smoothing a probability $1 - s$ is assigned to the true label w_j, and a probability of s (smoothing factor) is distributed uniformly among the rest of the k words in the vocabulary. We augment the distribution of the incorrect class by weighting the smoothing factor s according to the cosine similarity between the Glove (Pennington et al., 2014) word embedding of the correct word in the training data and all the words in the vocabulary. Thus, if the correct word to be predicted is "good", then the words "great" and "awesome" in the vocabulary would get a higher proportion of the smoothing factor s, compared to an unrelated word like "aeroplane"- presenting more accurate knowledge to the model. Mathematically, let $\vec{w_j}$ be the Glove word embedding of word w_j, $\mathbf{W_k}$ be a matrix containing the Glove word embedding for all the words in the vocabulary (including w_j), $\vec{w}_{j\_sim}$ be the vector of cosine similarity between the word w_j and all the words in the vocabulary. Since Glove word embeddings are learned representations, they can be noisy. Hence, we introduce a binary mask $mask_j$ using a threshold t, below which we set the cosine similarity value in $\vec{w}_{j\_sim}$ as 0, and multiply the similarity vector with the mask. The resulting vector is normalised to lie between 0 and 1, and finally multiplied by s. We treat t as a model hy-

[1]Code and dataset available here.

perparameter, and is tuned using grid search. We further reason that although Glove embeddings are learned from text corpora, there are possibilities that dissimilar words can lie in close proximity in the embedding space, resulting in a high cosine similarity score, and presenting an incorrect knowledge to the model. To circumvent this problem, we further experiment with filtering out the cosine similarities of dissimilar words based on WordNet sysnets (Miller, 1995), which we achieve by implementing another mask $mask_j^{'}$.

$$\vec{w}_{j\_sim} = \frac{\vec{w}_j \cdot \mathbf{W_k}}{||\vec{w}_j|| \cdot ||\mathbf{W_k}||} \tag{1}$$

$$\vec{w}_{j\_dist} = \frac{\vec{w}_{j\_sim} * mask_j * mask_j^{'}}{\sum(\vec{w}_{j\_sim} * mask_j * mask_j^{'})} * s \tag{2}$$

$$\vec{w}_{j\_dist}[j^*] = 1 - s \tag{3}$$

$$\text{where, } mask_j = \begin{cases} 0, & \text{if } \vec{w}_{j\_sim} <= t \\ 0, & \text{if } \vec{w}_{j\_sim} = 1 \\ 1, & \text{otherwise} \end{cases}$$

$$\text{and, } mask_j^{'} = \begin{cases} 1, & \text{if } w_k \text{ is a synonym of } w_j \\ 0, & \text{otherwise} \end{cases}$$

3.1 Dataset

We experiment with (i) The DailyDialog dataset (Li et al., 2017): A multi-turn open domain dialogue dataset comprising 13,118 conversations pertaining to diverse day-to-day topics, and (ii) The Empathetic Dialogues dataset (Rashkin et al., 2019): An open domain multi-turn dataset consisting of 25,000 conversations grounded in emotional situations. We use the same training, validation and testing splits as mentioned in the datasets. We concatenate all the turns in the query in one long text, and use two special tokens: *[speaker1]* and *[speaker2]* to distinguish the speakers. In order to speed up computation, we restrict the context to the most recent 50 tokens, which is determined analytically from the corpora. Additional training details and code in Section A.2 (Appendix A).

3.2 Model

Since the primary scope of this paper is to experiment with different loss functions, we used a standard transformer encoder-decoder architecture as proposed by Vaswani et al. (2017), where the encoder encodes the most recent utterance in the conversation, along with context from the previous turns. The encoder-decoder comprises of 3 layers each, with 300 dimensional hidden representation, with 6 attention heads in each multi-headed attention layer. The embedding layer is populated with 300 dimensional Glove embeddings, which are trained along with the entire network. Finally, a fully connected linear layer predicts the next word.

3.3 Experiments

We treat the Cross Entropy (CE) loss, CE loss with label smoothing, Kullback–Leibler (KL) divergence loss and KL loss with label smoothing as baselines. We experiment with different smoothing values $s \in \{\text{NA}, 0.1, 0.2\}$, cosine similarity thresholds $t \in \{\text{NA}, 0.0, 0.5, 0.8\}$, and also perform ablation study to analyze the usefulness of the WordNet similarity mask $w \in \{\text{NA}, 0, 1\}$. Overall we experiment with 30 diverse settings per dataset.

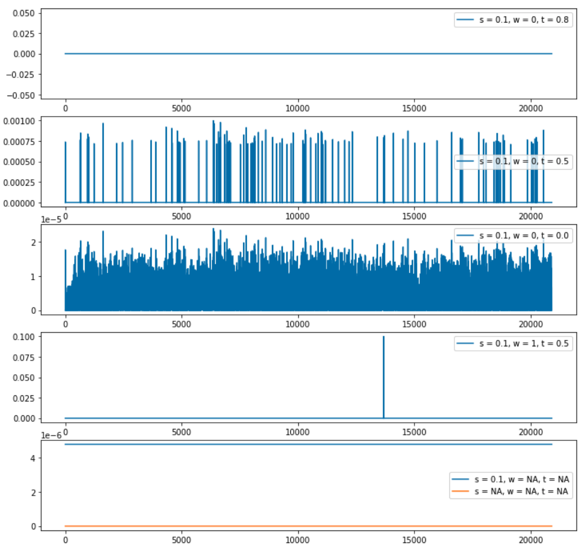


Figure 2: Illustration of incorrect word probabilities(x-axis = vocabulary, y-axis = probability). Setting t = 0.8, or using WordNet mask filters out most words making the target distribution equivalent to vanilla CE loss targets. Using t = 0.5 or 0.0 yields a less dramatic effect and preserves the information of the incorrect labels.

4 Results and Analysis

We compare the (i) sacreBLEU score (Post, 2018): a standardised version of the BLEU score (Papineni et al., 2002), (ii) ROUGE L score (Lin, 2004): which compares Longest Common Subsequence (LCS), and automatically takes into account sentence level structure similarity and identifies longest co-occurring in sequence n-grams, (iii) METEOR score (Banerjee and Lavie, 2005): an improvement over BLEU score, which incorporates stemming and synonymy matching along

			s = NA	**s = 0.1**	**s = 0.2**	**s = 0.1**				**s = 0.2**			
			t = NA	**t = NA**	**t = NA**	**t = 0**		**t = 0.5**		**t = 0**		**t = 0.5**	
Dataset	**Metric**	**Loss**	**w = NA**	**w = NA**	**w = NA**	**w = 0**	**w = 1**	**w = 0**	**w = 1**	**w = 0**	**w = 1**	**w = 0**	**w = 1**
DD	SacreBLEU	CE	1.662	1.852	1.725	1.989	1.762	**2.193 (+ 12.67%)**	1.857	2.115	1.802	2.053	1.930
		KL	**1.946**	1.753	1.793	1.845	1.818	1.912	1.885	1.938	1.809	1.729	1.885
	ROUGE L	CE	0.120	0.124	0.124	0.123	0.120	**0.127 (+ 0.57%)**	0.126	0.121	0.120	0.123	0.124
		KL	**0.126**	0.122	0.123	0.122	0.126	0.124	0.124	0.122	0.125	0.123	0.123
	METEOR	CE	0.124	**0.132**	0.128	0.134	0.128	**0.137 (+ 4.16%)**	0.131	0.134	0.127	0.134	0.131
		KL	**0.132**	0.130	0.130	0.134	0.132	0.132	0.131	0.131	0.131	0.129	0.129
ED	SacreBLEU	CE	2.279	**2.408**	2.190	**2.442 (+ 1.42%)**	2.208	2.192	2.216	2.318	2.262	2.312	2.256
		KL	2.271	2.168	2.279	2.277	2.278	2.337	2.361	2.431	2.274	2.439	2.143
	ROUGE L	CE	0.138	**0.143**	0.137	0.144	0.140	0.142	0.138	0.141	0.139	0.141	0.138
		KL	0.140	0.139	0.142	0.144	0.145	0.143	0.138	**0.146 (+ 1.95%)**	0.140	0.143	0.139
	METEOR	CE	0.125	0.128	0.125	**0.132 (+ 2.08%)**	0.126	0.124	0.124	0.129	0.126	0.127	0.124
		KL	0.125	0.123	**0.129**	0.127	0.130	0.129	0.124	0.132	0.125	0.128	0.123

Table 1: Comparison of sacreBLEU, ROUGE L and METEOR scores using variants of Cross Entropy (CE) loss and Kullback–Leibler (KL) divergence loss on DailyDialog (DD) and EmpatheticDialogues (ED) datasets; s denotes amount of smoothing, where s ∈ (0.1, 0.2, NA); t = cosine similarity threshold, where t ∈ (0, 0.5, NA); w = apply synonym based filtering, where w ∈ (0, 1, NA).

with exact word matching. Table 1 summarizes our results, where the columns containing "NA" are the baseline results, against which improvements are measured. Further, Section A.1 (Appendix A) contains results for all configurations with additional evaluation metrics like BERTscore (Zhang* et al., 2020) and ROUGE 1 & 2.

Observations From the experiments we observe that, **(i)** Using a data dependent cosine similarity based distribution for label smoothing significantly outperforms the baseline (vanilla entropy based loss with or without label smoothing). We observe 12.67 % increase in BLEU score, 0.57 % increase in ROUGE L score, and 4.16 % increase in METEOR score for the DailyDialog dataset, and 1.42 % increase in BLEU score, 1.95 % increase in ROUGE L score, and 2.08 % increase in METEOR score for the EmpatheticDialogues dataset. **(ii)** Using additional WordNet synonym based filtering (w) does not help performance. To understand why this is happening, we plotted the distribution of the smoothing factor s for the randomly selected word "fun", and observed that the word had only one overlapping WordNet synonym in our vocabulary: "play". This caused the word "play" to be assigned a probability of 0.1, while all the other words are assigned a probability of 0, except for "fun", which was assigned a probability of 0.9. We reason that the sparsity in synonyms does not help in reducing the overconfidence of the model, as the final distribution is very similar to non-smoothing targets. Figure 2 illustrates the probabilities assigned to the incorrect labels of the word "fun", by each of the methods discussed in this paper. **(iii)** Using CE loss instead of KL generally improves performance while using label smoothing. We reason that this happens because in case of label smoothing, the constant entropy coefficient in KL loss reduces the overall loss, thus reducing the gradients during back propagation, which results in slower learning. **(iv)** Generally, using high smoothing value (s) does not help in learning. **(v)** The cosine similarity threshold t should be treated as a hyperparameter, and will require tuning depending on the vocabulary of the dataset used. **(vi)** We also noticed that a cosine similarity threshold t as high as 0.8 does not help in learning. We reason that using a high threshold creates a scenario similar to using WordNet synonyms, where the smoothing probability is distributed among very few (or no) words. Note that in order to enhance readability, the results with 0.8 threshold are omitted from Table 1, and are presented in the additional supplementary materials.

5 Conclusion

Label smoothing has an undesirable property of assigning uniform probability to incorrect labels, which present an incorrect knowledge to learn from. In this paper we propose ways to convert the uniform distribution to a data dependent distribution by weighing the smoothing probability using cosine similarity of word embeddings between the correct and incorrect labels. We further experiment with WordNet synonyms as an additional filtering criteria, and report our findings. Although we achieve significant improvements over all baselines, we notice a drawback where the proposed system is unable factor in context while weighing the distribution of the incorrect labels. As future research, we intend to address this drawback using more contextualised representations instead of static embeddings.

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.

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. *arXiv preprint arXiv:2004.02594*.

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.

Tianxing He and James Glass. 2020. Negative training for neural dialogue response generation.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network.

Shaojie Jiang, Pengjie Ren, Christof Monz, and Maarten de Rijke. 2019. Improving neural response diversity with frequency-aware cross-entropy loss. *The World Wide Web Conference on - WWW '19*.

Daniel Kang and Tatsunori Hashimoto. 2020. Improved natural language generation via loss truncation.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. DailyDialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 986–995, Taipei, Taiwan. Asian Federation of Natural Language Processing.

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.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

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.

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.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing neural networks by penalizing confident output distributions.

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.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2019. Towards empathetic open-domain conversation models: A new benchmark and dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5370–5381, Florence, Italy. Association for Computational Linguistics.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the inception architecture for computer vision.

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

Guoyin Wang, Chunyuan Li, Jianqiao Li, Hao Fu, Yuh-Chen Lin, Liqun Chen, Yizhe Zhang, Chenyang Tao, Ruiyi Zhang, Wenlin Wang, Dinghan Shen, Qian Yang, and Lawrence Carin. 2020. Improving text generation with student-forcing optimal transport.

Yida Wang, Yinhe Zheng, Yong Jiang, and Minlie Huang. 2021. Diversifying dialog generation via adaptive label smoothing. 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 3507–3520, Online. Association for Computational Linguistics.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

A Appendix

A.1 All Experiment Results

Table 2 shows the different variants of the baselines that were computed for both the DailyDialog and EmpatheticDialogues datasets. All performance improvements are compared against these baselines. For a metric, the best baseline score among all the hyperparameter settings is chosen to report improvements. Table 3 shows the results of using different hyperparameter settings and loss function in the DailyDialog dataset, and Table 4 shows the results obtained on the EmpatheticDialogues dataset. The best results with detailed comparison against baselines are already discussed in the main paper.

A.2 Model Training and Parameters

All the models were trained on a single Nvidia V-100 GPU, for 15 epochs each with a learning rate of 2e-4, batch size of 64, and using AdamW optimizer. The gradients of the model were clipped with a value of 1, and dropout with probability 0.1 was applied during training. The average runtime of each experiment is 60 minutes, with each of the trained models having 17.7 M parameters. The code, dataset and best performing models are publicly available through this link: download link.

		DailyDialog Dataset			EmpatheticDialogue Dataset		
		s = NA	s = 0.1	s = 0.2	s = NA	s = 0.1	s = 0.2
		t = NA	t = NA	t = NA	t = NA	t = NA	t = NA
Metric	**Loss**	w = NA	w = NA	w = NA	w = NA	w = NA	w = NA
sacreBLEU	CE	1.6625	1.8523	1.7251	2.2794	2.4084	2.1903
	KL	1.9469	1.7536	1.7931	2.2715	2.1682	2.2797
BERTScore	CE	0.8522	0.8520	0.8520	0.8539	0.8544	0.8527
	KL	0.8529	0.8520	0.8510	0.8540	0.8531	0.8541
ROUGE 1	CE	0.1272	0.1312	0.1319	0.1536	0.1592	0.1527
	KL	0.1336	0.1298	0.1300	0.1560	0.1545	0.1587
ROUGE 2	CE	0.0282	0.0303	0.0299	0.0251	0.0292	0.0251
	KL	0.0305	0.0283	0.0282	0.0267	0.0259	0.0271
ROUGE L	CE	0.1209	0.1243	0.1243	0.1382	0.1437	0.1373
	KL	0.1263	0.1223	0.1233	0.1406	0.1395	0.1426
METEOR	CE	0.1244	0.1324	0.1286	0.1254	0.1287	0.1257
	KL	0.1324	0.1303	0.1303	0.1250	0.1233	0.1297

Table 2: Baseline results of diverse automatic text generation metrics on the DailyDialog and EmpatheticDialogues datasets. The hyperparameters s, t and w control the usage of Label Smoothing, Cosine similarity threshold and WordNet filtering respectively. For the baseline, t and w were not used, which is indicated by NA. s = NA signifies vanilla entropy based loss without Label Smoothing.

		s = 0.1						s = 0.2					
		t = 0		t = 0.5		t = 0.8		t = 0		t = 0.5		t = 0.8	
Metric	**Loss**	w = 0	w = 1	w = 0	w = 1	w = 0	w = 1	w = 0	w = 1	w = 0	w = 1	w = 0	w = 1
sacreBLEU	CE	1.9896	1.7627	2.1936	1.8575	1.6676	1.8859	2.1158	1.8020	2.0536	1.9302	1.5674	1.8502
	KL	1.8459	1.8181	1.9128	1.8858	1.7957	1.7453	1.9387	1.8092	1.7292	1.8856	1.5874	1.9707
BERTScore	CE	0.8518	0.8529	0.8515	0.8527	0.8507	0.8509	0.8513	0.8525	0.8525	0.8519	0.8507	0.8512
	KL	0.8520	0.8527	0.8517	0.8515	0.8520	0.8516	0.8509	0.8525	0.8522	0.8518	0.8518	0.8515
ROUGE 1	CE	0.1309	0.1279	0.1353	0.1326	0.1260	0.1280	0.1298	0.1271	0.1315	0.1317	0.1250	0.1290
	KL	0.1301	0.1332	0.1318	0.1311	0.1281	0.1276	0.1301	0.1328	0.1310	0.1312	0.1263	0.1325
ROUGE 2	CE	0.0282	0.0276	0.0309	0.0300	0.0287	0.0280	0.0286	0.0286	0.0308	0.0310	0.0276	0.0305
	KL	0.0283	0.0312	0.0300	0.0294	0.0297	0.0291	0.0285	0.0312	0.0292	0.0299	0.0277	0.0299
ROUGE L	CE	0.1238	0.1209	0.1270	0.1260	0.1200	0.1203	0.1217	0.1204	0.1238	0.1244	0.1183	0.1222
	KL	0.1227	0.1264	0.1243	0.1242	0.1213	0.1207	0.1223	0.1253	0.1232	0.1234	0.1185	0.1252
METEOR	CE	0.1342	0.1287	0.1379	0.1314	0.1270	0.1319	0.1344	0.1279	0.1346	0.1313	0.1223	0.1280
	KL	0.1346	0.1324	0.1327	0.1311	0.1262	0.1275	0.1319	0.1310	0.1298	0.1296	0.1247	0.1330

Table 3: Results of diverse automatic text generation metrics on the DailyDialog dataset, trained with variants of Entropy based loss with different hyperparameter settings: cosine similarity threshold (t), Label Smoothing (s) and WordNet filtering (w).

		s = 0.1						s = 0.2					
		t = 0		t = 0.5		t = 0.8		t = 0		t = 0.5		t = 0.8	
Metric	**Loss**	w = 0	w = 1	w = 0	w = 1	w = 0	w = 1	w = 0	w = 1	w = 0	w = 1	w = 0	w = 1
sacreBLEU	CE	2.4427	2.2082	2.1922	2.2164	2.3467	2.2596	2.3187	2.2622	2.3125	2.2569	2.3944	2.2767
	KL	2.2774	2.2781	2.3370	2.3615	2.2347	2.2769	2.4319	2.2749	2.4393	2.1431	2.2566	2.2652
BERTScore	CE	0.8543	0.8539	0.8547	0.8528	0.8536	0.8544	0.8544	0.8536	0.8539	0.8532	0.8531	0.8544
	KL	0.8541	0.8543	0.8544	0.8528	0.8536	0.8528	0.8544	0.8528	0.8544	0.8526	0.8535	0.8543
ROUGE 1	CE	0.1612	0.1564	0.1577	0.1531	0.1558	0.1551	0.1589	0.1550	0.1575	0.1531	0.1553	0.1590
	KL	0.1594	0.1613	0.1596	0.1540	0.1549	0.1552	0.1619	0.1554	0.1588	0.1545	0.1564	0.1569
ROUGE 2	CE	0.0287	0.0270	0.0271	0.0250	0.0274	0.0267	0.0269	0.0265	0.0267	0.0264	0.0261	0.0262
	KL	0.0270	0.0290	0.0273	0.0266	0.0256	0.0253	0.0288	0.0269	0.0274	0.0257	0.0251	0.0268
ROUGE L	CE	0.1443	0.1409	0.1425	0.1385	0.1402	0.1388	0.1416	0.1398	0.1411	0.1381	0.1396	0.1423
	KL	0.1441	0.1454	0.1435	0.1387	0.1397	0.1393	0.1465	0.1401	0.1430	0.1394	0.1404	0.1416
METEOR	CE	0.1324	0.1266	0.1248	0.1245	0.1267	0.1264	0.1291	0.1266	0.1278	0.1243	0.1247	0.1292
	KL	0.1272	0.1302	0.1290	0.1246	0.1235	0.1254	0.1323	0.1253	0.1283	0.1234	0.1257	0.1269

Table 4: Results of diverse automatic text generation metrics on the EmpatheticDialogues dataset, trained with variants of Entropy based loss with different hyperparameter settings: cosine similarity threshold (t), Label Smoothing (s) and WordNet filtering (w).

A Novel Approach towards Cross Lingual Sentiment Analysis using Transliteration and Character Embedding

Rajarshi Roychoudhury[1], **Subhrajit Dey**[1], **Md Shad Akhtar**[2]
Amitava Das[3], **Sudip Kumar Naskar**[1]
[1]Jadavpur University, India
[2]IIIT-Delhi, India
[3]AI Institute, University of South Carolina, Columbia, SC, USA
{rroychoudhury2,subhrajitdey.agt,sudip.naskar}@gmail.com, shad.akhtar@iiitd.ac.in, amitava@mailbox.sc.edu

Abstract

Sentiment analysis with deep learning in resource-constrained languages is a challenging task. In this paper, we introduce a novel approach for sentiment analysis in resource-constrained scenarios using character embedding and cross-lingual sentiment analysis with transliteration. We use this method to introduce the novel task of inducing sentiment polarity of words and sentences and aspect term sentiment analysis in the no-resource scenario. We formulate this task by taking a metalingual approach whereby we transliterate data from closely related languages and transform it into a meta language. We also demonstrated the efficacy of using character-level embedding for sentence representation. We experimented with 4 Indian languages – Bengali, Hindi, Tamil, and Telugu, and obtained encouraging results. We also presented new state-of-the-art results on the Hindi sentiment analysis dataset leveraging our metalingual character embeddings.

1 Introduction

Sentiment analysis is a widely explored topic in the field of Natural Language Processing, which focuses on classifying text into 3 sentiment classes: positive, negative, and neutral (Liu, 2012). For any text classification task, supervised approaches require an extensive and domain specific corpus in the corresponding language. However, sentiment annotated data, which is the primary resource for sentiment analysis, is limited in many languages. Transfer learning can be used to learn sentence representations by pretraining on a large corpus and finetuning to a particular downstream task. But transfer learning often fails in adapting knowledge from one domain to another, and sufficient training samples across these domains may not be available for efficient finetuning. Another method to deal with such situations is to create cross lingual tools between a resource rich and resource poor language. This method either uses machine translation between the language or bilingual dictionaries to overcome the language gap. However, it is an extremely challenging task to create an accurate machine translation system between two languages or create a bilingual dictionary pair in a resource constrained scenario (Balamurali et al., 2012). Moreover, there are some studies (Lohar et al., 2017, 2018; Pal et al., 2014; Kumari et al., 2021) that show that there is a significant loss of pragmatics in the translations produced by the state-of-the-art Machine Translation (MT) systems, which could adversely affect the performance of these downstream NLP applications that use unedited raw MT output. Very little work has been done to use cross-lingual sentiment analysis without translation involved in any step, and even if they do, they require the presence of large sentiment annotated data in at least one language.

Another major problem is the unavailability of significantly large corpus to train the language-specific word embedding models in resource scarce languages. As a result, training word embeddings on those languages and reusing the pretrained embeddings is not feasible. Literature suggests that using character embedding or subword embedding can be an efficient alternative, as the number of unique characters and alphabets for any language are well deterministic (Chrupała, 2013; dos Santos and Gatti, 2014; Mikolov et al., 2012; Kim, 2019). The main advantage of using character embedding is that when we encounter a word which is not in the domain of the available corpus (i.e., an out-of-vocabulary word, or OOV), we can still have a latent representation of the word for any NLP task. Previously, dos Santos and Gatti

(2014) used character embedding for sentiment analysis of short texts, however, very few work have been done on word level classification.

There exist many such languages where the coverage of the corpus is too poor to effectively train the embedding weights for each word. Even transfer learning with pretrained large models like m-BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), Electra (Clark et al., 2020) might encounter many unknown ([UNK]) tokens since the vocabulary set of the tokenizers may not have many words in the test-set. Moreover, a dataset for the downstream task may not exist in the language for effective finetuning. Therefore, we need methods that can help create sentiment lexicons effectively and perform sentiment analysis without using any training dataset of the target language.

In this paper, we introduce a novel method to learn efficient latent representation by projecting data in multiple languages to a shared metadata representation. We propose to utilize the WX-notations as the transliteration approach for the projection. This enables us to leverage the collective training corpus to learn a deep learning system with higher confidence. We evaluate our approach on 7 sentiment analysis datasets (4 word-level, 2 sentence-level, and 1 aspect-level) across four Indian languages – Hindi, Telugu, Bengali, and Tamil. Moreover, we experiment in two setups – mono-lingual and cross-lingual. Finally, we compare our model against various baselines and observe the performance to be better for the majority of the cases. Convolutional Neural Network (CNN) provides the best result on aspect analysis and sentiment classification of sentences, while different models prove to be useful on different datasets for word-level sentiment analysis.

Contribution: The paper makes the following contributions.

- We introduce the novel task of predicting the sentiment polarity of words.
- Word-level and sentence level *zero-shot* sentiment analysis using a metalingual approach.
- Our proposed approach provides new state-of-the-art results on the $Review_{SH}$, $Review_{AH}$ and $Movie_{H}$ (Akhtar et al., 2016) Hindi sentiment analysis datasets.

2 Related Work

Wan (2009) introduced the method of cross-lingual sentiment analysis, which leveraged an available English corpus for Chinese sentiment classification by using the English corpus as training data without using any Chinese resources using a co-training approach. Zhou et al. (2016) proposed a joint learning algorithm that exploits both monolingual and bilingual constraints, where the monolingual constraints help to model words and documents in each language while the bilingual constraints help to build a consistent embedding space across languages. Abdalla and Hirst (2017) used cross sentiment analysis by computing a matrix to convert from the vector space of one language to that of another, based on the fact that that sentiment is highly "preserved" even if translation accuracy is poor. Here it is worth noting that most of these methods use a single resourceful language and use it for a resource-scare language, what sets us apart is that we have used multiple source languages and combined their resources to train hybrid embedding weights.

Rasooli and Collins (2017) combined a method for deriving cross-lingual clusters and a method for transfer of lexical information from the target language into source language treebanks with the density-driven approach to annotation projection for cross-lingual sentiment analysis on different source and destination languages instead of a single source language. Jain and Batra (2015) used Bilingually Constrained Recursive Auto-encoder (BRAE) (Zhang et al., 2014) to perform Cross-Lingual sentiment analysis. However, in most of the works cited above, we have to use translation to obtain cross-lingual relations between language pairs. As mentioned earlier, statistical machine translation is computation-intensive and demands an extensive corpus of bilingual text.

The main difficulty is cross alignments, due to word order/ syntactic differences in languages. Balamurali et al. (2012) presented an alternative approach to CLSA using WordNet senses as features for supervised sentiment classification. But for many resource-constrained languages, WordNet does not exist. Hence in the proposed methodology, we propose using transliteration as alignments are monotonic, ie, they do not cross each other. Previously an attempt was made to transliteration for sentiment analysis. In oder to automatically classify sentiments of Arabizi messages. Guellil et al. (2018) transliterated their corpus into Arabic and used classification models like Support Vector Machines, Naive Bayes classifier and Decision Trees for sentiment analysis. However for our experiments we will use a transliteration scheme to obtain a common metalanguage that can be used cross-lingual embedding training, and classify words/ sentences using simple classification models without using any data of the resource-scarce language.

3 Proposed Method

Training Deep neural network requires huge amount of data, which is not feasible for low resource languages. To avoid this, we follow the principles of cross-lingual learning, where model trained on one language is reused on another language. This ensures that no data from the target language is used during training, and hence can be adapted in a low resource scenario.

Our approach is as follows: similar sentiment-annotated datasets from other languages are leveraged to create a pool of data instead of a single source language data, which is subsequently transliterated into a common language (metalanguage) using a transliteration scheme. We use this combined metadata as our training data, which acts as a relatively bigger dataset that can be used for efficient training of deep learning-based text classification models. Additionally, it reduces the dependency on a single language dataset. The target language dataset is also transliterated using the same transliteration scheme and this transliterated data is used only as the testing data and plays no part in training.

We train our classification models on the transliterated data from other languages, and use the trained classification models to classify words/sentences of the transliterated test set of the target language. Thus we build our sentiment classification model for a language without using any training data of that language or involving any translation. As shown in the later sections, this method yields in comparable results given by state-of-the-art models like BERT in text classification in scarce-resource scenarios where very little or no training data is available.

3.1 Transliteration - WX notation

We use WX notation (Gupta et al., 2010) – a transliteration scheme for representing Indian languages in ASCII. In this transliteration scheme, every consonant and vowel has a single mapping into Roman. Hence it is a prefix code, advantageous from a computation point of view. In the WX notation, typically lower case letters are used for unaspirated consonants and short vowels while capital case letters are used for aspirated consonants and long vowels. While the retroflexed voiceless and voiced consonants are mapped to ‘t’, ‘T’, ‘d’, and ‘D’, the dentals are mapped to ‘w’, ‘W’, ‘x’, and ‘X’. Hence the name of the scheme, WX, refers to the idiosyncratic mapping.

3.2 Word and Sentence Representation

One question that naturally arises is how to represent the words and sentences and what embedding should be used. Our experiments aim to create sentiment lexicon (word level) and classify text (sentence level). Because of this, combined with the fact that there is no corpus in such meta language, the natural choice is using character embedding. Character embedding also provides the benefit that we can have a numeric representation of the new words that get coined to the language. Following dos Santos and Gatti (2014), given a word W composed of M characters $\{c_1, c_2, .r.., c_M\}$, we first transform each character c_m into a character embedding r_m^{chr} . Character embeddings are encoded by column vectors in the embedding $W^{chr} \in \mathrm{R}^{d^{chr} \times |V^{chr}|}$. Given a character c, its embedding r^{chr} is obtained by the matrix-vector product $r^{chr} = W^{chr} v^c$, where v^c is a vector of size $|V^{chr}|$, which has value 1 at index c and zero in all other positions. Thus, W is represented by the sequence of character embeddings $\{r_1^{chr}, r_2^{chr}, r_3^{chr}, \ldots r_M^{chr}\}$.

After transliteration, we label encode each character of this metalanguage and represent words as a vector of these labels (c.f. Figure 1). Each character is one-hot encoded, with the length of the vector set to the number of unique characters in this language. Each of

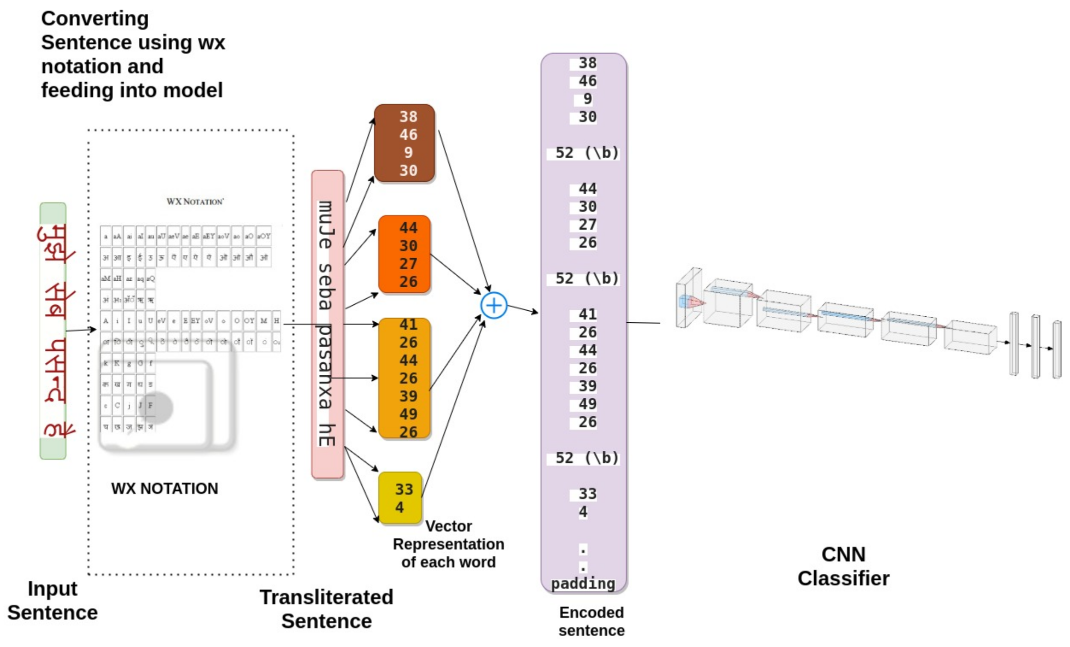


Figure 1: Encoding sentence

these vectors is used as the initial embedding weights for each character. For sentences, the representation differs slightly. Sentences are also represented as a vector of characters, with the difference that an extra character is inserted into the vector which represents space. The embedding weight of this character is initialized as 0, representing a NULL vector. We carried out several experiments to test and compare how the method works under different scenarios, and how to simulate the various situations where this method can be of use, the details of which are given in Section 5.

4 DataSet

We used the SentiWordNet[1] for Indian Languages (Das and Bandyopadhyay, 2010a; Das and Gambäck, 2012; Das and Bandyopadhyay, 2011) for our experiments. This dataset contains sentiment polarity of words in four Indian languages – Bengali (BN), Hindi (HN), Tamil (TA) and Telugu (TE). For each language, the dataset contains four different files LC_POS[2], LC_NEG, and LC_NEU and LC_AMBI, listing the set of positive, negative, neutral, and ambiguous words, respectively. LC_AMBI was not used in our experiments on word-level classification due to the negligible presence of ambiguous words compared to the rest of the classes. Each word in this dataset is marked with POS category[3]. Table 1 presents the statistics of the dataset.

For the actual sentiment analysis task, we used the Aspect Sentiment dataset ($Review_{AH}$), Review Sentiment dataset ($Review_{SH}$) and Movie Review Sentiment dataset ($Movie_{H}$)[4] (Akhtar et al., 2016). These 3 Hindi datasets are annotated with 4 sentiment classes – positive (POS), negative (NEG), neutral (NEU), and conflict (CON). Table 2 presents the statistics of these datasets.

Language	POS	NEG	NEU	Total
Bengali	1,779	3,714	359	5,852
Hindi	2,313	2,337	371	5,801
Telugu	2,136	4,076	359	6,571
Tamil	2,225	4,447	361	7,033

Table 1: Statistics of the SentiWordNet for Indian Languages

[1]https://amitavadas.com/sentiwordnet.php
[2]LC is the language code – BN: Bengali, HN: Hindi, TA: Tamil, TE: Telugu.
[3]a: Adjective, n: Noun, r: Adverb, v: Verb, u: unknown
[4]www.iitp.ac.in/~ai-nlp-ml/resources.html

Dataset	POS	NEG	NEU	CON	Overall
$Review_{SH}$	2,290	712	2,226	189	5,417
$Movie_{H}$	823	530	598	201	2,152
$Review_{AH}$	1,986	569	1,914	40	4,509

Table 2: Statistics of the Hindi Sentiment Analysis Datasets

5 Experimental Setup

For the classification task, we used various models – RNN, CNN, LSTM, GRU, Self Attention Mechanism (Letarte et al., 2018) and a combination thereof. Embedding length in each model is set to the number of unique characters. For the metalanguage experiments, since the WX format converts characters to uppercase and lowercase English characters, the embedding size is set to 52. Each convolution layer has embedding dimension=1, number of input channel=embedding length, kernel size=1, and number of output feature =3. For LSTM and GRU, the embedding dimension is equal to the input size which in turn is equal to the embedding length, number of hidden states=512, and a dropout rate of 0.01. Each model ended with a dense layer with a softmax activation layer for classification. Many of these layers were combined one on top of the other for sentiment classification of an individual dataset. It was tested with the following learning rates: 0.001,0.005,0.05,0.5 and the best results have been reported. Since the data suffers from class imbalance, proportionate samples were taken from each class. For stacked layers, the hyperparameters were tweaked accordingly to adjust input and output dimensions. For the metalanguage experiments, Adam optimizer was used with a learning rate of 0.001 and sparse categorical cross-entropy as the loss function.

6 Experiments and Results

We conducted experiments in 2 directions – (i) inducing sentiment polarity of words, and (ii) sentence-level sentiment analysis and aspect analysis.

6.1 Sentiment Polarity Prediction of Words

We carried out two sets of experiments for inducing sentiment polarity in words – (a) using training data from the target language, and (b) without using training data from the target language.

We first applied word sentiment classification to individual datasets by using character embedding and different deep learning-based models, namely CNN, LSTM, GRU, Self Attention, and a combination of them. Table 3 presents the results of these experiments.

Different models (along with different learning rates) seem to work better for different languages. For the monolingual setup, GRU, CNN, BiGRU and Self Attention produced the best performance for Bengali, Hindi, Tamil, and Telugu, respectively. Hindi proves to be a challenging language for this task. The best accuracy obtained for Hindi is 51.25 while for the rest 3 languages the accuracy varies in the range [60, 65]. If all the languages and models are considered, CNN with LSTM or GRU layer performs consistently better than the other models.

Following our proposed methodology, for sentiment polarity induction of words in a new language (e.g., TE), we converted both the target language (i.e., TE) and other language (i.e., HN, BN, TA) datasets to a common metalanguage using the wx notation. Then we combined all the other language (i.e., HN, BN, TA) datasets to form the training set and trained our classification models on this dataset. The trained models were then used to predict the sentiment polarity for the target (i.e., TE) dataset. Thus in this experiment, we simulated the scenario where we used no training data of the target language itself, instead used sentiment annotated datasets from other languages for training.

For the cross-language experiments, Self-Attention, CNN+GRU, LSTM, and CNN produced the best results on BN, HN, TA, and TE, respectively. Interestingly, HN proves to be a challenging language for the cross-lingual setup as well.

Table 5 summarizes the best results for each language for both monolingual and cross-

Architecture	Accuracy(%)							
	Monolingual				Cross-Lingual			
	BN	HN	TA	TE	BN	HN	TA	TE
LSTM	64.10	49.17	63.81	61.61	56.53	47.04	**63.76**	60.03
GRU	**65.00**	48.25	62.90	62.16	56.28	47.64	62.80	61.03
BiLSTM	63.10	48.50	62.86	62.27	54.69	49.78	63.08	61.97
BiGRU	64.70	46.75	**64.09**	60.83	56.45	49.60	58.14	57.80
CNN	63.70	**51.25**	63.86	62.38	57.11	49.95	61.80	**62.31**
CNN+LSTM	64.30	50.00	62.86	62.50	56.25	49.75	60.86	61.53
CNN+GRU	64.50	49.00	64.00	62.11	56.33	**50.22**	60.52	60.81
Self Attention	62.70	47.75	62.86	**62.83**	**63.41**	46.96	63.20	62.03

Table 3: Results of Word Sentiment Polarity Identification using Monolingual and Cross-lingual Frameworks

Architecture	Accuracy(%)					
	Using training data			Without Using training data		
	$Review_{SH}$	$Movie_H$	$Review_{AH}$	$Review_{SH}$	$Movie_H$	$Review_{AH}$
Akhtar et al. (2016)	57.34	44.88	65.96	-	-	-
m-BERT	62.69	51.04	59.96	-	-	-
CNN	**61.34**	**46.51**	**68.63**	42.07	**38.38**	**43.88**
LSTM	55.89	42.12	63.30	**42.27**	37.19	42.06
RNN	52.98	42.89	60.86	41.86	33.90	41.48
CNN+LSTM	57.28	43.41	64.04	39.49	36.71	40.11
CNN+GRU	54.28	43.93	64.75	31.47	37.92	43.56
BiGRU	54.50	40.05	62.08	39.36	33.13	39.28
GRU	55.01	40.83	62.64	42.20	34.22	40.66
Self-Attention	53.90	40.13	60.31	37.65	34.72	43.15

Table 4: Results of Sentence-Level Sentiment Analysis and Aspect Analysis

lingual frameworks. It can be observed from Table 5 that the results obtained in the cross-lingual framework are typically lower in accuracy than the results obtained with the monolingual framework, which is quite expected, however, the scores are not very far away. In the absence of any training data for the target language, the results of the cross-lingual experiments can be considered significant.

Train	**Test**	**Architecture**	**Accuracy**
BN	BN	GRU	65.00
TE+HN+TA		Self Attention	63.41
HN	HN	CNN	51.25
BN+HN+TA		CNN+GRU	50.22
TA	TA	BiGRU	64.09
BN+HN+TE		LSTM/GRU	63.76
TE	TE	Self Attention	62.83
BN+HN+TA		CNN	62.31

Table 5: Best results on each testset for the monolingual and metalingual frameworks

6.2 Sentiment Analysis of Sentences and Aspect Analysis

We used the $Review_{SH}$ and $Movie_H$ Hindi datasets (Akhtar et al., 2016) for sentence level sentiment classification and $Review_{AH}$ for aspect term sentiment analysis. Sentences from all these 3 datasets were transliterated from Hindi to the metalanguage with the wx notation. We used the Bengali, Tamil, and Telugu datasets of SentiWordNet (Das and Bandyopadhyay, 2010b) to train the character embeddings which are subsequently used in the classification models. Like the word level sentiment polarity prediction (cf. Section 6.1), we considered two scenarios to classify the sentences – (a) using the sentence-level training data, and (b) without using the sentence-level training data. In both the techniques the language in which the dataset is originally built (Hindi in our case) is not directly used for training the model. Instead, the sentences are transliterated to the wx notation and encoded using the technique specified earlier.

80% of the dataset was used as training data and 20% was treated as the testset. Evaluation results are reported in Table 4 under the column "Using training data". We received the best accuracy of 61.34, 46.51, and 68.63 on the $Review_{SH}$, $Movie_H$ and $Review_{AH}$ datasets, respectively, which are significantly better than the results reported in (Akhtar et al., 2016). CNN produced the best results across all the datasets. We also fine-tuned BERT on the datasets using multilingual-BERT(m-BERT) as an encoder followed by a simple classification header. Although BERT outperforms in most cases, our scores are still comparable to those obtained with BERT.

For the case where sentence level training data is not used, we did not use any portion of the $Review_{SH}$, $Movie_H$ and $Review_{AH}$ datasets for training. Instead, we used transfer learning where the model that was trained to determine the embedding weights of each character and classify words is used to classify the sentences. It is to be noted that we did not even use the Hindi dataset of SentiWordNet to train the character embeddings for this experiment. The evaluation results are shown in Table 4 under the column "Without using training data". As expected, the obtained results are much lower than the corresponding results reported under column "Using training data" in Table 4. However, the results suggest that in a resource-constrained scenario where there is no training data available, this method can act as an effective way of classification.

7 Analysis of Results

Results suggest that our proposed method performs reasonably well for different tasks and languages. The traditional CNN architectures captured structural features very well, which is evident from the fact that sentiment embedded vectors when incorporated in training produced state-of-the-art results on most of the datasets. We realize that using CNN will give us two main advantages: (i) learn hidden semantics from a metalanguage, and (ii) handling limited coverage of lexical resources. Even when training data is not used, we achieved promising results which proves the effectiveness of the method in a resource-constrained scenario. An interesting observation was found in the $Review_{AH}$ dataset, where the model gave different results based on the convolution window size. For example, the models performed better when the sentence comprised of the aspect term and four words from either side of the aspect term and used for sentiment classification than the situation when 2 words from either side were considered for training. This can be attributed to the fact that the information from the distant part of the sentences is sometimes attributed to the overall sentiment polarity of the aspect terms. However, the situation reversed when no training data was used, where sentences with 2 words from either side provided better classification results in comparison to sentences with 4 words from either side of aspect terms. This is because we used word-level classification models to classify these sentences, and hence these models captured the local features of the aspect terms more efficiently and gave better accuracy.

8 Conclusions

In this paper, we introduced the novel task of inducing the sentiment polarity of words using character embedding-based deep learning models. We extended the task to inducing the sentiment polarity of words in a new language having no training data. We carried out experiments with 4 Indian languages and obtained encouraging results. The cross-lingual approach proved to be an effective method in a resource-constrained scenario. The same idea was followed to perform sentiment analysis and aspect analysis in Hindi without using any training data in Hindi. While using training data, our method outperformed the previous state-of-the-art in sentiment analysis and aspect analysis in Hindi.

References

Mohamed Abdalla and Graeme Hirst. 2017. Cross-lingual sentiment analysis without (good) translation. *arXiv preprint arXiv:1707.01626*.

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.

AR Balamurali, Aditya Joshi, and Pushpak Bhattacharyya. 2012. Cross-lingual sentiment analysis for indian languages using linked wordnets. In *Proceedings of COLING 2012: Posters*, pages 73–82.

Grzegorz Chrupała. 2013. Text segmentation with character-level text embeddings. *arXiv preprint arXiv:1309.4628*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Amitava Das and Sivaji Bandyopadhyay. 2010a. Sentiwordnet for indian languages. In *Proceedings of the Eighth Workshop on Asian Language Resouces*, pages 56–63.

Amitava Das and Sivaji Bandyopadhyay. 2010b. SentiWordNet for Indian languages. In *Proceedings of the Eighth Workshop on Asian Language Resouces*, pages 56–63, Beijing, China. Coling 2010 Organizing Committee.

Amitava Das and Sivaji Bandyopadhyay. 2011. Dr sentiment knows everything! In *Proceedings of the ACL-HLT 2011 System Demonstrations*, pages 50–55, Portland, Oregon. Association for Computational Linguistics.

Amitava Das and Björn Gambäck. 2012. Sentimantics: Conceptual spaces for lexical sentiment polarity representation with contextuality. In *Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis*, pages 38–46, Jeju, Korea. 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.

Cícero dos Santos and Maíra Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Imane Guellil, Ahsan Adeel, Faical Azouaou, Fodil Benali, Ala-eddine Hachani, and Amir Hussain. 2018. Arabizi sentiment analysis based on transliteration and automatic corpus annotation. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 335–341, Brussels, Belgium. Association for Computational Linguistics.

Rohit Gupta, Pulkit Goyal, and Sapan Diwakar. 2010. Transliteration among indian languages using wx notation. In *KONVENS*.

Sarthak Jain and Shashank Batra. 2015. Cross lingual sentiment analysis using modified brae. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 159–168.

Gyuwan Kim. 2019. Subword language model for query auto-completion. *arXiv preprint arXiv:1909.00599*.

Divya Kumari, Asif Ekbal, Rejwanul Haque, Pushpak Bhattacharyya, and Andy Way. 2021. Reinforced NMT for sentiment and content preservation in low-resource scenario. *ACM Trans. Asian Low Resour. Lang. Inf. Process.*, 20(4):70:1–70:27.

Gaël Letarte, Frédérik Paradis, Philippe Giguère, and François Laviolette. 2018. Importance of self-attention for sentiment analysis. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 267–275, Brussels, Belgium. Association for Computational Linguistics.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

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*.

Pintu Lohar, Haithem Afli, and Andy Way. 2017. Maintaining sentiment polarity in translation of user-generated content. *The Prague Bulletin of Mathematical Linguistics*, 108(1):73.

Pintu Lohar, Haithem Afli, and Andy Way. 2018. Balancing translation quality and sentiment preservation (non-archival extended abstract). In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 81–88.

Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky. 2012. Subword language modeling with neural networks. *preprint (http://www. fit. vutbr. cz/imikolov/rnnlm/char. pdf)*, 8(67).

Santanu Pal, Braja Gopal Patra, Dipankar Das, Sudip Kumar Naskar, Sivaji Bandyopadhyay, and Josef van Genabith. 2014. How sentiment analysis can help machine translation. In *Proceedings of the 11th International Conference on Natural Language Processing*, pages 89–94.

Mohammad Sadegh Rasooli and Michael Collins. 2017. Cross-lingual syntactic transfer with limited resources. *Transactions of the Association for Computational Linguistics*, 5:279–293.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 235–243.

Jiajun Zhang, Shujie Liu, Mu Li, Ming Zhou, and Chengqing Zong. 2014. Bilingually-constrained phrase embeddings for machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 111–121.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2016. Cross-lingual sentiment classification with bilingual document representation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1412.

Normalization of Spelling Variations in Code-Mixed Data

Krishna Yadav[1], Md Shad Akhtar[1], Tanmoy Chakraborty[2]
[1]IIIT Delhi, India; [2]IIT Delhi, India.
{krishna19039, shad.akhtar}@iiitd.ac.in, tanchak@ee.iitd.ac.in

Abstract

Code-mixed text infused with low resource language has always been a challenge for natural language understanding models. A significant problem while understanding such texts is the correlation between the syntactic and semantic arrangement of words. The phonemes of each character in a word dictates the spelling representation of a term in low resource language. However, there is no universal protocol or alphabet mapping for code-mixing. In this paper, we highlight the impact of spelling variations in code-mixed data for training natural language understanding models. We emphasize the impact of using phonetics to neutralize this variation in spelling across different usage of a word with the same semantics. The proposed approach is a computationally inexpensive technique and improves the performances of state-of-the-art models for three dialog system tasks *viz.* intent classification, slot-filling, and response generation. We propose a data pipeline for normalizing spelling variations irrespective of language.

1 Introduction

There are around 6,500 languages spoken in the world today(Wikipedia contributors, 2021). *English, Mandarin, Chinese* tops the list with over 2 billion speakers around the globe hence are highly resourceful languages. On the other hand, there are resource-scarce languages such as *Polish, Odia, Hindi and, many more* with few million speakers only. Due to lack of resources understanding such languages poses a great challenge for the research community. Natural Language Understanding (NLU) means extracting the semantic schema of the utterance to re-act according to the intent of the utterance. NLU is crucial for any human-to-machine interaction-based system such as chatbots, virtual assistants, and many more. Now the mode of communication is restricted by the speaker's language and innate understanding of language systems.

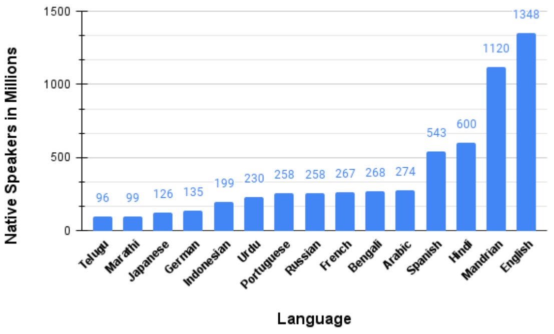


Figure 1: Native Speakers Count

The Goal-Oriented Dialogue System (Young, 2000) was first introduced based on dialog state tracking (Williams et al., 2013) and gave a new direction to the NLU tasks. A number of datasets are available in diverse domains for like-wise downstream goal-oriented conversational data such as ATIS (Hemphill et al., 1990), SNIPS (Coucke et al., 2018), DSTC (Williams et al., 2013), WOZ (Budzianowski et al., 2018), etc. However, all of them are monolingual, i.e., available in the English language only. Code-mixing is the form of linguistics where the conveyer uses two (or more than two) languages together so that some words of the low resource language replace the words from high resource language or vice versa. The usual trend is to mix English with any other regional language such as Hindi (Hindi + English $\rightarrow$ Hinglish), Bengali (Bengali + English $\rightarrow$ Benglish), Tamil (Tamil + English $\rightarrow$ Tamilish), and many more. With the increase in multi-lingual speakers, code-mixing is very common in online platforms, social media, and day-to-day life (Gumperz, 1982; Gysels, 1992; Durán, 1994; Moyer, 2002). Most common activities such as shopping, restaurant reservations, booking tickets, and so on all involve extensive use of Code-mixing. For example, a Hindi speaking

Language	Utterances
English	**Speaker 1**: Hi, Can you help me in booking a table at this restaurant? **Speaker 2**: Sure, would you like something in cheap, moderate or expensive price range?
Hinglish	**Speaker 1**: Hi, kya tum is restaurant mein ek table book karne mein meri help karoge? **Speaker 2**: Sure, kya aap cheap, moderate ya expensive price range mein kuch like karenge?

Table 1: Sample Code-Mixed Utterances.

user looking to book a restaurant would typically ask, ”Kya tum is restaurant mein ek table book karne mein meri help karoge?” (”Can you help me in booking a table at this restaurant?”) (Banerjee et al., 2018).

A significant proportion of the population worldwide is using code-mixed language over online platforms (Singh et al., 2018). The prime complication with linguistic diversity is such that there is no convention or protocol to refer to when it comes to code-mixing. Hence, it depends on the writer’s perception and knowledge of the phonics of the source language. (Kukich, 1992) grouped writing errors into two classes. First one is typographical that occurs when a character is substituted by the wrong character whose key is nearby in the keyboard or interchanging of the character order, for instance, *merw paas(mere paas), kimd(kind), kys krna ha(kya krna hai), and more variations due to different reasons.* Other class is of cognitive errors that occur when the writer is unaware of the native spelling and semantics of that word. In this case, the wrongly spelled word is phonetically close to the correct word (Toutanova and Moore, 2002). We assume that the chances of typographical errors are less since the writer intends to avoid making such errors. This paper mainly focuses on cognitive errors and covers a major portion of typographical errors as well.

In the case of code-mixed data, there are no rules that can lead towards achieving the correct spelling because there is no correct spelling. We can assume the most commonly used representation as correct and normalized text to get some contextual meaning. The introduction of external knowledge can also help to improve the results of spelling correction. We propose a computationally inexpensive novel technique to normalize spelling variations irrespective of the language of the bilingual speaker.

2 Related Work

Divergence from the traditional spelling and having variation for the same word often carry some meaning (Sebba, 2007). In computational linguistics, while dealing with digital forms of regional text forms, it becomes helpful to map all spelling variations (semantically identical) to the same point in embedding space. (Nguyen and Grieve, 2020) highlighted a detailed analysis on the same. They analyzed that the skip-gram model, which does not consider spelling variations, encode spelling variation patterns to some extent. Also, the use of cosine similarities helped find a link between intentional variations and distance from the conventionally followed standards.

Historical writings face the identical problem of high degree variance in spellings (Reynaert et al., 2012)—every day with new findings in historical text and extending the corpora in digital form (literature). Various researches already explored the normalization approaches based on string distance measures to a reasonable extent for proposing various tools for normalization. (Reynaert et al., 2012) shows that, individually, the rule-based method (Norma Tool) performed best in the presence of a large amount of training data (Bollmann et al., 2012). A combination of normalization methods produces the best results and helps in further cleaning and processing of data. Hence, integrating simple word-to-word mappers always increase the overall performance. Methods like Edit Distance or Levenshtein distance (Levenshtein, 1965; Yujian and Bo, 2007) needs a massive corpus of universally correct data. (Bollmann and Søgaard, 2016) further gave improvisation for this problem with the use of bi-LSTM network (Schuster and Paliwal, 1997) applied on a character level. Multi-task learning with additional normalization (integration of mappers) improves the model’s performance. Their model outperformed the CRF-based models and Norma tool given in (Bollmann et al., 2012). Extending the work keeping in mind the idea of integrating, (Domingo and Casacuberta, 2018) proposed three approaches based on statistical, neural, and character-level machine translation to train the model concerning modern spelling variation standards. Their model covered a holistic view of the word-to-word mappers. Additionally, they proposed a simplistic approach of a statistical dictionary, similar to a word-to-word mapper in which they used the changing frequency of spelling on the training corpora. They also stated that the statistical machine translation approach gave better results than neural machine translation on small corpora. (Lertpiya et al., 2020) explored another low resource language which was (*Thai*). In their work, they proposed a two-staged pipeline with neural contextual attention. Using neural error corrector and Seq2Seq error corrector alleviates the problem of *overcorrection*.

The phonetically motivated approach has also been explored a little by (Downs et al., 2020) where they prioritized the phonetic key of the misspelled word over supplementary ones. A survey conducted for (Weld et al., 2021) reports an excellent survey on joint intent classification and slot filling techniques. They NLU models of over a decade and gave a detailed comparison with the pros and cons of various techniques. Concluding the state-of-the-art research, they provide multiple comparisons that best summarise the past work done along different dimensions, including the features, base approaches, and dataset domain used. Hybrid phonetic neural models (Viana-Cámara et al., 2021) and BERT (Devlin et al., 2018) models have also been explored to capture character-phonetic but they don't capture the code-mixed data. (Hládek et al., 2020) conducted a survey of spelling correction techniques. They studied the interactive process of error production and correction. All the major research assumes that the correct spelling of the miss-spelled word is native to the written language. In the case of code-mixed data, no such thesaurus exists. Additionally, there is the absence of any particular set of rules that one can use for code-mixing, and it is complex to come up with such a method or protocol to translate one language into a romanized language.

Recently, (Sengupta et al., 2021) came up with a method for sub-word level representation learning that is supplemented by the word level lexical variations in code-mixed languages. They evaluated the proposed architecture on a mix of European and Indic languages *(Spanish, Hindi, Bengali, Tamil, Telugu, and Malayalam)*. They proposed a Hierarchically attentive Transformer (HIT) framework, a novel architecture to encode text semantic and syntactic features in an embedding space with efficacy. It learns word representations at the sub-word level using a Fused Attention MEchanism (FAME). It incorporates two major attention components. An outer product attention (OPA) (Le et al., 2020) to extract higher-order character-level similarities and multi-headed self attention (MSA) (Vaswani et al., 2017), a standard transformer module that computes a scaled query-key vector pair dot product. FAME extends the MSA module by including OSA and calculates their weighted sum. The proposed model tries to embed semantically and phonetically similar words of a code-mixed language by capturing relevant information at a more granular level but lacks overall coverage of spelling variations. HIT is computationally expensive with over trainable parameters *2.7M* for sequence classification and over *1.4M* for POS tagging. It misses the essence of layman language in text utterances.

3 Dataset Used

We will use a code-mixed version of the DSTC2 dataset (Williams et al., 2013; Henderson et al., 2014a,b; Williams et al., 2014, 2016). They incorporated code-mixing in four regional languages Hindi, Bengali, Gujarati, and Tamil, romanized as English (Banerjee et al., 2018) by crowd-sourcing the data for language translation from native speakers of respective languages. The data contains 50k utterances on the restaurant reservation domain, including getting reservations done or asking for information such as restaurant address, phone, etc. Final data contains bot-to-human dialog conversations. The authors converted the raw data from audio format to text. For this task, the authors used Automatic Speech Recognition (ASR) modules.

Sentence	cheap	restaurant	south	west	mein
Slots	B-Price	O	B-Area	I-Area	O
Intent	inform				

Table 2: BIO-Tagging

Representation: There are 3 slots, 5 possible areas, 91 cuisines, and 3 price ranges. Workers acting as customers were requested to deviate the conversation in the middle of the dialog to various slots and their possible values to make data robust, less intuitive, and unconditional and avoid unnecessary patterns in data.

The workers transcribed the conversations and labeled the utterances with different dialog states. For example, each utterance was labeled with its semantic intent representation (request[area], inform[area = north]) and the dialog turns were labeled with annotations such as constraints on the slots (cuisine = Italian), requested slots (requested = phone, address) and the method of search (by constraints, by alternatives). Such annotations are useful for domain-specific slot-filling based dialog systems (Banerjee et al., 2018). This whole process consists of three phases. Extracting unique utterances from DSTC2, i.e., dialogues with only change in the slot values rest same are filtered out. Creating code mixed translations using Amazon Mechanical Turk (AMT) tool for crowd-sourcing and by in-house workers. The evaluation involved taking random dialogues from the dataset for colloquialism (unforced translation), Intelligibility (easily understandable), and Coherence(irrespective of neighboring utterances knowledge).

The quantitative measure of code-mixing : As per the evaluation process. The authors analyzed the obtained code-mixed data for code-mixing measures. (Gambäck and Das, 2016) gave a metric to measure code-mixing in a sentence given as:

$$: if N(x) > 0$$

$$C_u(x) = 100.\frac{N(x) - \max\limits_{L_i \in \mathcal{L}}\{t_{L_i}\} + P(x)}{2N(x)} \quad (1)$$

Here, the measure of code-mixing for sentence x, $C_u(x)$ is given by $N(x)$, number do foreign language tokens in sentence. Maximum number of tokens t in language L_i from the set of languages $\mathcal{L}$. In addition, number of language switch points given by $P(x)$. In the above equation, the language of the majority of words in the sentence serve as *Embedding* however, irrespective of the majority, we need to consider English as *Embedding* and Hindi as *Matrix* language. To over come this the authors proposed to replace the general $\max\limits_{L_i \in \mathcal{L}}\{t_{L_i}\}$ with $native(x)$ given as:

$$native(x) = \begin{cases} \{t_{L_n}\} & : t_{L_n} > 0 \\ N(x) & : t_{L_n} = 0 \end{cases} \quad (2)$$

Here, t_{L_i} is the number of tokens in native language(Hindi). Also, the term $\delta(x_i)$ with values 0(if switch in Matrix language) or 1(purely English) to measure the extent of inter-utterance code-mixing and frequency. The authors also considered the fraction of code-mixed utterances $\frac{S:CMUtterances}{U:TotalUtterances}$. And the final equation is given as $C_u(x)$ for one utterance and $C_c(x)$ for complete corpus:

Property	Count
Total Utterances	49167
Unique utterances	6733
Utterances per dialogue	15
Words per utterance	8
Words per dialogue	120
KB triples per dialogue	38
Train dialogue	1168
Validation dialogue	500
Test dialogue	1117
Vocab Size	1229

Table 3: Raw Dataset Analysis.

Property	Count in Hinglish
Unique Utterances	6549
Code-Mixed Utterances	5750
Hindi Only Utterances	348
English Only Utterances	451
Utterances per dialogue	12
Words per utterance	8
Hindi Vocab Size	739
English Vocab Size	551
Code-Mixed Vocab Size	386

Table 4: Code-Mixed Dataset Analysis.

$$\begin{aligned} C_u(x) &= \left(1 - \frac{native(x) + P(x)}{N(x)} + \delta(x)\right) \\ C_c(x) &= \frac{100}{U}\left[\frac{1}{2}\sum_{i=1}^{U} C_u(x) + \frac{5}{6}S\right] \end{aligned} \quad (3)$$

4 Proposed Methodology

We propose a novel method of robust systems for low resource similar token spelling variations. The main reason for little spelling variations in the pronunciation and ambiguous phonetics of the word. Due lack of any formal conversion criterion, the writer introduces variations. The aim is to map the semantically similar romanized words such as *Kabhi, kabi, kbhi, etc* together so that they can be neutralized to a single term and have identical embedding. Then give a most common and close to

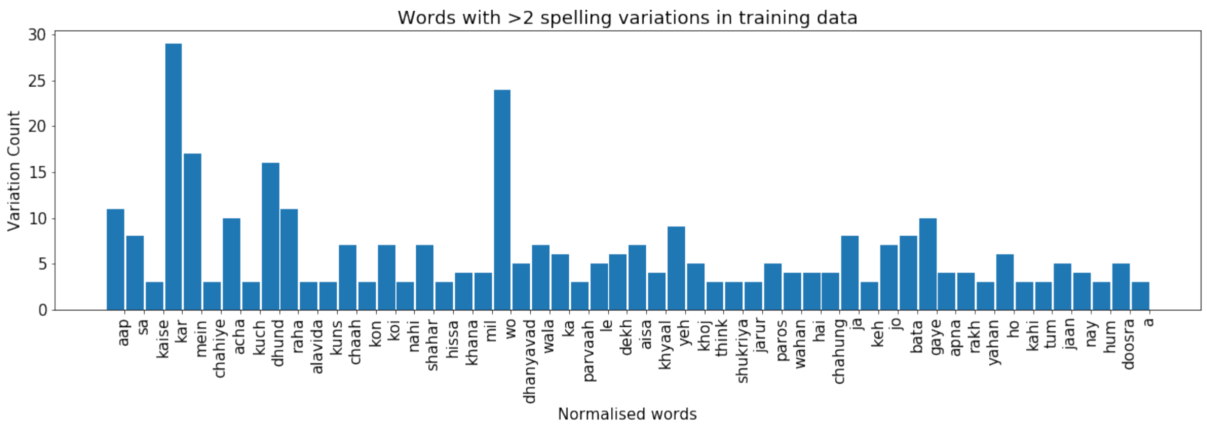


Figure 2: Spelling Variation Count Analysis

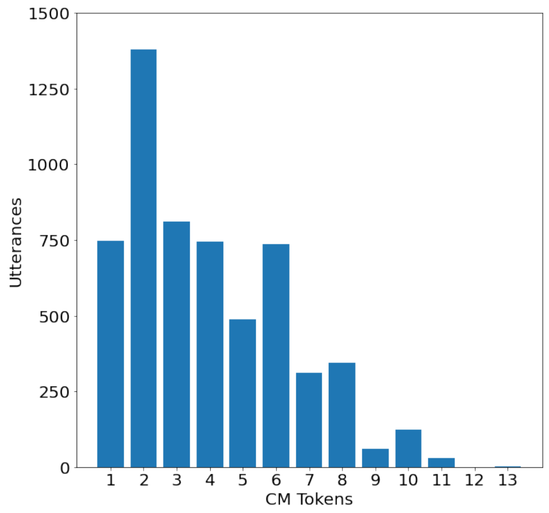


Figure 3: Utterance Count VS Hindi word count

correct representation of that word. We approach this statement in 5 phases as follows:

1. **Capture Code Mixing:** The point of focus is the tokens that belong to *Matrix* Language. For this, we markdown the predefined tokens (*here, intent labels, slot labels* and special tokens) according to the training data. For this purpose, we use English Dictionary for identifying the embedded language words. We leave the English words unchanged and sent the non-English words to the second phase.

2. **CM Elocution:** In this phase, we try to find a set of possible transliterations of each token in their native script *here, Devanagari*. We have an option to take the transliterated terms as to be syntactically and semantically incorrect in the *Matrix* Language since the whole idea is to capture the different pronunciation styles in the native language of the word. We use *indic-transliterator* (Bhat et al., 2015) for this job and stored top five closest transliterations for each romanised token. These transliterations sound similar to each other, with a minor change in terms of vowels and consonants.

3. **Candidate Selection (Devanagari Phoneme):** The set of recently formed *Devanagari* tokens are the closest possible phonetically similar terms that all sound the same but differ in the writing style and hence are close to each other. To reduce variation, we need to normalize all the possible variations with the most commonly used term. Now that we have to query for Hindi (*Devanagari*) text, there is plenty of corpora that we can take as a benchmark. It may or may not be semantically correct, but it will be the most used term by the majority of the population. We used the IIT Bombay parallel corpus (Kunchukuttan et al., 2017) to perform candidate selection. We use TF-IDF on the *Devanagari* translation of the dataset. The term with the max score is selected to be the best possible normalization for all the remaining terms. This newly elected normalized term which syntactically correct as per the *Matrix* Language. This way, we are able to pull the spelling variations together in multi-dimensional embedding space of the *Matrix* Language.

4. **Romanisation:** This step is similar to the 2^{nd} phase. For the CM dataset, the Devanagari terms are converted to their romanized

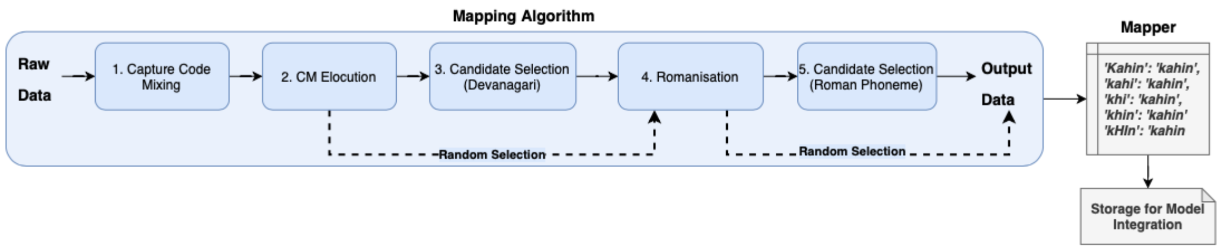


Figure 4: Proposed Data Pipeline.

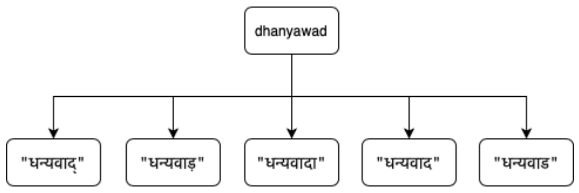


Figure 5: Devanagari Transliteration Generation

elocution by transliteration. Now, this is the crucial step; we do not have any order or protocol for elocution until now. However, we intentionally introduce minor spelling variations as part of code-mixing noise. Now when we have the map of code-mixed words, we can normalize it again based on candidate selection, and this way, we normalize the whole set of writer introduced code-mixing to abide by the code-mixing methodology.

Count	Language	Train	Test	Dev
Sentences		1,492,827	2,507	520
Tokens	English	20,667,259	57,803	10,656
	Hindi	22,171,543	63,853	10,174

Table 5: IIT Bombay Corpus Statistics

5. **Candidate Selection (Romanised Phoneme):** This step is similar to the earlier candidate selection, the only difference being the language to the terms. Here, we select one most used romanized *matrix* language term. TF-IDF scores catered to this selection from a rich bi-lingual corpus collected from various social media platforms to capture the latest trends of code-mixing generalized to various writers. We used Facebook, Twitter, WhatsApp chat dataset given by (Das, 2016) further explored in (Ramesh and Kumar, 2016). This dataset is enriched with a quality code-mixing that best caters to our need to figure out the most widely used representations of a *matrix* language term. With this, we get a single normalized term for all the syntactic variations of semantically similar hi-English terms. This output contains non-English terms without spelling variations for roman English (Hindi) words.

With this, we give novel methods and pre-determined mapping for most commonly used spelling variations with their preferred normalization. This mapping can be helpful in learning models for spelling correction for natural language understanding tasks.

5 Experiments

We perform numerous trials and runs to prove the importance of spelling normalization. We took a downstream task of intent classification and slot prediction. We use the BIO-Tagging format to transform the data for performing joint learning. We then compare the performance of state-of-the-art algorithms on our normalized data and old data with variations.

1. **CS-ELMO** (Aguilar and Solorio, 2019) used a state-of-the-art monolingual model for finding the sentence embedding for dialogue systems and used transfer learning to develop a model for code-switched bilingual text. Their model transfer English knowledge from a pre-trained ELMo model to code-switched language (Hi-English) using the task of language identification (Matrix language and Embedding Language). They used character convolutions for capturing character positions of unknown words. Their model outperformed the multilingual BERT (Devlin et al., 2018), and another code-switching ignorant monolingual model like ELMO (Peters et al., 2018).

2. **Stack Propagation** (Qin et al., 2019) considered the strong correlation of intent classification and slot filling (Zhang and Wang, 2016;

Hakkani-Tür et al., 2016; Liu and Lane, 2016). Qin's model used the intent information directly into the slot filling stage. Once the utterance's intent is classified, we concatenate the label to the slot representation for predicting the slots for each of the tokens simultaneously. They performed token level intent classification for further alleviating error propagation and finally passed the representation to the BERT layer for further performance gain. The semantic knowledge in the form of intents of each utterance act as a differential link between the two tasks. They also demonstrated the use of gated architecture proposed in (Goo et al., 2018).

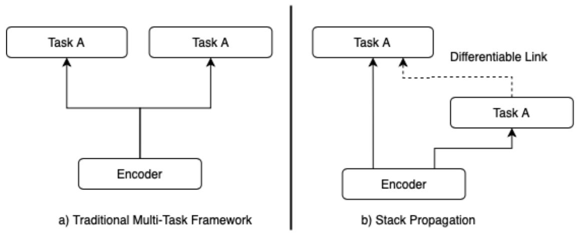


Figure 6: Multi-Task Vs Stack Propagation

3. **DCA-Net** Recently, (Qin et al., 2021) demonstrated an excellent work of propagating semantic knowledge from one task to another in the form of joint learning. In (Qin et al., 2019) there was a unidirectional flow of the information to overcome this Qui et al. further extended their work and proposed a co-interactive transformer module by establishing a bidirectional cross-impact between the two tasks in a unified architecture. Both intent classification and slot filling can take advantage of mutual information. Their model achieved state-of-the-art performance. The major drawback of their model is that it fails to incorporate code-switching and fails on our Hi-English data. We can further tune this method for extending the scope for multilingual data.

4. **HIT** A robust representation learning method was recently proposed by (Sengupta et al., 2021). They computed the weighted sum of two attention modules, multi-headed self-attention, and an outer product attention module, to obtain the final attention weights. An outer product attention (OPA) (Le et al., 2020) to extract higher-order character-level similarities and multi-headed self attention (MSA) (Vaswani et al., 2017), a standard transformer module that computes a scaled query-key vector pair dot product. HIT was able to encode syntactic and semantic features in embedding space by learning sub-word level representations with their fused mechanism called *FAME*. FAME extends the MSA module by including OSA and calculates their weighted sum. This model did not perform well due to incompetencies of efficiently reducing the distance between semantically same but syntactically varied code-switched terms.

6 Implementation

We report BLEU-4 (Papineni et al., 2002) and ROUGE-1, ROUGE-2 and ROUGE-L scores (Lin, 2004) for natural language generation machine translation task to compare the results with the dataset baselines given by (Banerjee et al., 2018; Banerjee and Khapra, 2019). Further, we compute Precision, Recall, and F1 scores for intent classification and Slot Filling evaluation. We calculate weighted and macro average scores and then chose macro overweighted because this is a class imbalance problem. The weighted average will give significant weightage to the most frequent class whose performance may lead to 100%. To encounter this issue, we report macro averaged scores, i.e., an average of independent scores for each class, treating all classes equally.

Further, We play with different combinations of the state-of-the-art algorithms and compared the performance on both syntactically normalized *(Spell)* and un-normalized *Inc_spell* versions of the dataset. We represent the data as each line containing a token and utterances separated by an empty line. Each token corresponds to a BIO tagging label, where B refers to the label's beginning, I refers to the intermediate words of the label, and O refers to other classes (no label of interest).

7 Results and Comparison

The table 7 shows the effect of normalizing spelling variations in code-mixed data and clearly shows that our novel architecture helps further to improve the quality of natural language generation tasks. The baseline algorithms Seq2Seq and Hred (Banerjee et al., 2018) along with graph convolutions network with sequential attention (Banerjee and

			Intent Macro Average				Slot Macro Average			
Model	**Data**	**Intent Acc**	**Pre**	**Rec**	**F1**	**Slot Acc**	**Pre**	**Rec**	**F1**	**Improvement**
CS-ELMo	Original	86.7	69.89	60.3	64.75	**99.55**	72.89	70.06	71.44	0.5464
CS-ELMo	Normalized	86.03	69.12	58.66	63.46	99.47	72.04	68.78	70.37	
Stack Prop	Original	93.56	86.51	78.89	81.98	98.85	76	76	77	0.3942
Stack Prop	Normalized	91.17	85.04	77.38	81.03	98.65	76	77	77	
cselmo+ stackProp	Original	**94.17**	**87.22**	**79.71**	**83.29**	99.23	**78.77**	78.38	**79.81**	0.4364
cselmo+ stackProp	Normalized	93.23	86.02	78.92	82.45	98.97	78.02	78.19	78.67	
DCA-Net	Original	85.73	75.3	62.7	66.86	98.92	69.21	63.64	64.08	0.2792
DCA-Net	Normalized	84.6	74.43	63.53	66.25	98.76	68.47	62.81	63.68	
cselmo+DCA-Net	Original	85.77	62.01	61.82	60.57	98.52	66.85	67.69	64.28	0.4771
cselmo+DCA-Net	Normalized	84.32	61.17	61.02	59.94	98.12	65.96	66.81	63.49	
HIT	Original	86.33	76.75	63.59	67.68	98.28	61.35	**84.39**	52.78	0.37217
HIT	Normalized	85.36	75.83	64.52	66.55	97.44	60.72	83.98	51.54	

Table 6: Comparing different combinations of state-of-the-art models on Hi-English Data.

Model	**Data**	**Bleu**	**Rouge-1**	**Rouge-2**	**Rouge-L**
Seq2Seq	Normalized	55.1	62.9	52.5	61
Seq2Seq	Original	55.9	63.55	53.1	62.09
Hred	Normalized	55.3	63.4	52.7	61.15
Hred	Original	55.61	63.92	53.25	61.91
GCN-SeA	Normalized	57.1	66.4	56.8	64.4
GCN-SeA	Original	57.4	66.78	56.4	65.98

Table 7: Effect of Spelling Normalization

Khapra, 2019) both results in higher performance increased by *0.5 - 1.5 units*, when normalised data is used. Normalization forces the vector representations for semantically identical but syntactically close terms to overlap each other in multidimensional embedding space and forces the model to treat all possible variations as the same only.

From table 6 we can infer, for all the state-of-the-art approaches, the proposed modification has lead to a significant performance gain by a factor of 0.5%-1.5%. The maximum improvement of 1.23 in slot F1 and 1.35 in intent F1 score can be seen in CS-ELMO, considering it explicitly focuses on the various types of character level embedding to capture context. The combinations of CS-ELMO with DCA-Net (0.8 for slot F1 and 0.67 in intent F1) and CS-ELMO with Stack Propagation (1.14 in slot F1 and 0.84 in intent F1) algorithms also shows significant performance improvement. Hence, the importance of tackling spelling variation in code mixed data is evident from the above results. Integrating different modules results in overcoming the drawbacks of each module in one way or another. This helps in improving the over performance.

8 Conclusion

With this emerging trend of code-mixing over social media platforms and daily communication in almost every region, the necessity to develop efficient natural language understanding models has increased. We communicate in a specific language (say, English) because the language has set standards, semantics, syntactic, and phonetics. Without any standard for low-resource languages, it is tough to communicate. Different people have different ways of pronouncing and depicting the alphabet, phonetics, and accent of a language close to their mother tongue. This action may lead to spelling variations while writing text as a medium for communication. We introduce a novel, computationally inexpensive, fully robust, and efficient method to normalize these spelling variations that work as an auto-correct to counter this problem. This mechanism helps in performance gain not only machine translation but also generic natural language tasks and any downstream task involving code-mixing problems. We give a universal mapping for Hi-English code-mixing that can be used directly by making a query in $O(1)$ time to normalize non-English words. It is evident from our work that integrating several modules together helps in performance improvement. We can further improve this solution by learning a model to further normalize out of the vocabulary terms.

Acknowledgment

Authors acknowledge the support of Infosys Center of Artificial Intelligence (CAI), IIIT Delhi.

References

Gustavo Aguilar and Thamar Solorio. 2019. From english to code-switching: Transfer learning with strong morphological clues. *arXiv preprint arXiv:1909.05158*.

Suman Banerjee and Mitesh M Khapra. 2019. Graph convolutional network with sequential attention for goal-oriented dialogue systems. *Transactions of the*

Association for Computational Linguistics, 7:485–500.

Suman Banerjee, Nikita Moghe, Siddhartha Arora, and Mitesh M Khapra. 2018. A dataset for building code-mixed goal oriented conversation systems. *arXiv preprint arXiv:1806.05997*.

Irshad Ahmad Bhat, Vandan Mujadia, Aniruddha Tammewar, Riyaz Ahmad Bhat, and Manish Shrivastava. 2015. Iiit-h system submission for fire2014 shared task on transliterated search. In *Proceedings of the Forum for Information Retrieval Evaluation*, FIRE '14, pages 48–53, New York, NY, USA. ACM.

Marcel Bollmann, Stefanie Dipper, Julia Krasselt, and Florian Petran. 2012. Manual and semi-automatic normalization of historical spelling-case studies from early new high german. In *KONVENS*, pages 342–350.

Marcel Bollmann and Anders Søgaard. 2016. Improving historical spelling normalization with bidirectional lstms and multi-task learning. *arXiv preprint arXiv:1610.07844*.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Inigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. Multiwoz–a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Amitava Das. 2016. Tool contest on pos tagging for code-mixed indian social media (facebook, twitter, and whatsapp) text. In *Proceedings of the 13th International Conference on Natural Language Processing*.

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*.

Miguel Domingo and Francisco Casacuberta. 2018. Spelling normalization of historical documents by using a machine translation approach. *Congresos - EAMT2018 - Proceedings*.

Brody Downs, Oghenemaro Anuyah, Aprajita Shukla, Jerry Alan Fails, Maria Soledad Pera, Katherine Wright, and Casey Kennington. 2020. Kidspell: A child-oriented, rule-based, phonetic spellchecker. *LREC 2020, Twelfth International Conference on Language Resources and Evaluation, 6937-6946*.

L. Durán. 1994. Toward a better understanding of code-switching and interlanguage in bilinguality: Implications for bilingual instruction. In *The journal of educational issues of language minority students*.

Björn Gambäck and Amitava Das. 2016. Comparing the level of code-switching in corpora. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1850–1855.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. 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 753–757.

John J Gumperz. 1982. *Discourse strategies*. 1. Cambridge University Press.

M. Gysels. 1992. French in urban lubumbashi swahili: Codeswitching, borrowing, or both? *Journal of Multilingual and Multicultural Development*, 13:41–55.

Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Interspeech*, pages 715–719.

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*.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014a. The second dialog state tracking challenge. In *Proceedings of the 15th annual meeting of the special interest group on discourse and dialogue (SIGDIAL)*, pages 263–272.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014b. The third dialog state tracking challenge. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 324–329. IEEE.

Daniel Hládek, Ján Staš, and Matúš Pleva. 2020. Survey of automatic spelling correction. *Electronics*, 9(10).

Karen Kukich. 1992. Techniques for automatically correcting words in text. *ACM Comput. Surv.*, 24(4):377–439.

Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. 2017. The iit bombay english-hindi parallel corpus. *arXiv preprint arXiv:1710.02855*.

Hung Le, Truyen Tran, and Svetha Venkatesh. 2020. Self-attentive associative memory. In *International Conference on Machine Learning*, pages 5682–5691. PMLR.

Anuruth Lertpiya, Tawunrat Chalothorn, and Ekapol Chuangsuwanich. 2020. Thai spelling correction and word normalization on social text using a two-stage pipeline with neural contextual attention. *IEEE Access*, 8:133403–133419.

V. Levenshtein. 1965. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics. Doklady*, 10:707–710.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454*.

Melissa G. Moyer. 2002. Pieter muysken, bilingual speech: A typology of code-mixing. cambridge: Cambridge university press, 2000. pp. xvi, 306. hb $ 59.95. *Language in Society*, 31:621 – 624.

Dong Nguyen and Jack Grieve. 2020. Do word embeddings capture spelling variation? In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 870–881.

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.

Libo Qin, Wanxiang Che, Yangming Li, Haoyang Wen, and Ting Liu. 2019. A stack-propagation framework with token-level intent detection for spoken language understanding. *arXiv preprint arXiv:1909.02188*.

Libo Qin, Tailu Liu, Wanxiang Che, Bingbing Kang, Sendong Zhao, and Ting Liu. 2021. A co-interactive transformer for joint slot filling and intent detection. In *ICASSP*.

Sree Harsha Ramesh and Raveena R Kumar. 2016. A pos tagger for code mixed indian social media text-icon-2016 nlp tools contest entry from surukam. *arXiv preprint arXiv:1701.00066*.

Martin Reynaert, Iris Hendrickx, and Rita Marquilhas. 2012. Historical spelling normalization. a comparison of two statistical methods: Ticcl and vard2. *Proceedings of ACRH-2*, pages 87–98.

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

Mark Sebba. 2007. *Spelling and society: The culture and politics of orthography around the world*. Cambridge University Press.

Ayan Sengupta, Sourabh Kumar Bhattacharjee, Tanmoy Chakraborty, and Md Shad Akhtar. 2021. Hit: A hierarchically fused deep attention network for robust code-mixed language representation. *arXiv preprint arXiv:2105.14600*.

Kushagra Singh, Indira Sen, and Ponnurangam Kumaraguru. 2018. Language identification and named entity recognition in hinglish code mixed tweets. In *Proceedings of ACL 2018, Student Research Workshop*, pages 52–58.

Kristina Toutanova and Robert C Moore. 2002. Pronunciation modeling for improved spelling correction. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 144–151.

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.

Rafael Viana-Cámara, Mario Campos-Soberanis, and Diego Campos-Sobrino. 2021. Hybrid phonetic-neural model for correction in speech recognition systems. *arXiv preprint arXiv:2102.06744*.

HENRY Weld, Xiaoqi Huang, SIQU Long, Josiah Poon, and SOYEON CAREN Han. 2021. A survey of joint intent detection and slot-filling models in natural language understanding. *arXiv preprint arXiv:2101.08091*.

Wikipedia contributors. 2021. List of languages by number of native speakers — Wikipedia, the free encyclopedia.

Jason Williams, Antoine Raux, and Matthew Henderson. 2016. The dialog state tracking challenge series: A review. *Dialogue & Discourse*, 7(3):4–33.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413, Metz, France. Association for Computational Linguistics.

Jason D Williams, Matthew Henderson, Antoine Raux, Blaise Thomson, Alan Black, and Deepak Ramachandran. 2014. The dialog state tracking challenge series. *AI Magazine*, 35(4):121–124.

Steve J Young. 2000. Probabilistic methods in spoken–dialogue systems. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 358(1769):1389–1402.

Li Yujian and Liu Bo. 2007. A normalized levenshtein distance metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1091–1095.

Xiaodong Zhang and Houfeng Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 2993–2999. AAAI Press.

A Method for Automatically Estimating the Informativeness of Peer Review

Prabhat Kumar Bharti[1], **Tirthankar Ghosal**[2], **Mayank Agarwal**[1], **Asif Ekbal**[1]
[1]Department of Computer Science and Engineering, Indian Institute of Technology Patna, India
[2] UFAL, MFF, Charles University, Czech Republic
prabhat_1921cs32@iitp.ac.in, ghosal@ufal.mff.cuni.cz
mayank265@iitp.ac.in, asif@iitp.ac.in

Abstract

Peer reviews are intended to give authors constructive and informative feedback. It is expected that the reviewers will make constructive suggestions over certain aspects, e.g., novelty, clarity, empirical and theoretical soundness, etc., and sections, e.g., problem definition/idea, datasets, methodology, experiments, results, etc., of the paper in a detailed manner. With this objective, we analyze the reviewer's attitude toward the work. Aspects of the review are essential to determine how much weight the editor/chair should place on the review in making a decision. In this paper, we used a publicly available Peer Review Analyze dataset of peer review texts manually annotated at the sentence level (13.22 k sentences) across two layers: Paper Section Correspondence and Paper Aspect Category. We transform these categorical annotations to derive an informativeness score of the review based on the review's coverage across section correspondence, aspects of the paper, and reviewer-centric uncertainty associated with the review. We hope that our proposed methods, which are motivated towards automatically estimating the quality of peer reviews in the form of informativeness scores, will give editors an additional layer of confidence for the automatic judgment of review quality. We make our codes available at https://github.com/PrabhatkrBharti/informativeness.git.

1 Introduction

The peer review process is the central mechanism for validating scientific research (Siler et al., 2015). A good review typically provides feedback on one or more sections and aspects while reviewing the manuscript/paper [1], rather than just one section, say the Introduction (Kühne et al., 2010). Therefore, reviews covering more sections and aspects are more likely helpful to the author. Furthermore, the more sections and aspects the review covers, the higher the expected coverage score. It may give the author a confidence that the reviewer has read through and paid attention to the different sections and aspects in their submission. In addition, the reviewers are expected to provide constructive comments and suggestions regarding certain aspects and sections of the manuscript. To determining whether the reviewer was informative or constructive in their review and covered significant sections of the manuscript. It would be appropriate to mention the data from Peer Review Analyze (Ghosal et al., 2022a). They analyze and understand the reviewers' thrust over specific sections and aspects of the manuscript. We use those insights in our proposed method. This particular motivation led us to incorporate the general sections, and aspects of the paper defined by the Peer Review Analyze (Ghosal et al., 2022a) into this paper to calculate the informativeness score. We attempt here to generate an informativeness score for a given review directly by analyzing the review's coverage across section correspondence, aspects of the paper, and reviewer-centric uncertainty associated with the review.

We summarize the key contributions of this work as follows.

- We propose a seed idea for the automatic judgment of review quality.
- We introduce a novel method for measuring the informativeness score based on sections, aspects coverage, and reviewer-centric uncertainty encapsulated in the review.
- In addition, we establish statistical-driven baselines to evaluate Mean absolute error (MAE), Root Mean Square Error (RMSE) and coefficient of determination (R^2).

The novelty of our work lies in utilizing the Peer

[1]In this manuscript, manuscript/paper are used interchangeably.

Review Analyze dataset for measuring the informativeness score. Although we use the reviews of a premier machine learning conference (ICLR) as our dataset, our proposed method would represent a generic aspect of peer review in Science, Technology, Engineering and Mathematics (STEM). It will assist the editors in which review they should pay more attention to when crafting a meta-review. In addition, it may give the author confidence that if the review has high informativeness score, it means the reviewer has reviewed thoroughly their submission.

2 Related Work

In the Meta Science community and Peer Review Congress[2] (Brezis and Birukou, 2020), peer review quality has been a major research topic since 1989. There are a few relevant ones that we discuss in this article. The authors (Justice et al., 1998) studied a randomized control trial to see how masking author identity improves peer review quality. The study in(Jefferson et al., 2002) presented approaches for assessing the quality of editorial peer reviews. To assess peer reviews of manuscripts, the authors of (Van Rooyen et al., 1999) developed the Review Quality Instrument (RQI). In this paper, the authors (Shattell et al., 2010) examined the perspectives of authors and editors on the quality of peer review in three scholarly nursing journals. Peer review quality is evaluated in (Van Rooyen, 2001). A systematic review and meta-analysis on the impact of interventions to improve the quality of peer reviews of biomedical journals were conducted in (Bruce et al., 2016). In this paper, authors (Enserink, 2001) explored the dubious connection between the peer review and quality. Authors (D'Andrea and O'Dwyer, 2017) argued if the editors can save peer reviews from peer reviewers. (Rennie, 2016) advocates scientific guidelines for peer review. The purpose of this (Callaham et al., 1998) study was to evaluate the reliability of the editor's opinion subjective quality ratings of peer review of manuscripts. This paper provides an overview of how peer-review reports of scientific articles can be assessed by the authors (Sizo et al., 2019). For peer reviews, some relevant NLP/ML works are worth exploring from an NLP/ML perspective (Kumar et al., 2021; Ghosal et al., 2019; Ghosal, 2019; Kumar et al., 2022; Ghosal et al., 2022b; Bharti et al., 2022a,b, 2021; Gao et al., 2019). It should be noted, however, that none of these works attempted to determine the quality of peer reviews based on linguistic aspects. Here, the goal is to derive a justifiable informativeness score and then use those insights to investigate further, enabling editors to automatically identify the quality of peer reviews.

3 Dataset

The dataset used in this study is from Peer Review Analyze (Ghosal et al., 2022a), which is publicly available. In Peer Review Analyze, peer review texts are manually annotated at the sentence level (13.22k sentences) across two layers: Paper Section Correspondence and Paper Aspect Category. The detailed dataset statistics are presented in Table 1, and the reader is referred to the original paper for further information.

3.1 Proposed Method

As we review the standard guidelines [3,4,5,6] for peer-reviewing in machine learning (ML) and natural language processing (NLP) conferences, we learn that the community expects a good review that covers more sections and aspects of the reviewed manuscript (Gregory and Denniss, 2019; Kühne et al., 2010). Having this motivation led us to develop a justifiable informativeness score which enables editors to automatically identify good reviews and isolate those that are less thorough. In our view, a good peer review should comment on key sections and highlight the reviewer's perspective while focusing on the essential aspects of the manuscript.

Peer Review Analyze dataset is used to generate an informativeness score based on the coverage of section correspondence, aspects of the paper, and the reviewer-centric uncertainty inherent in the review.

Paper Section Correspondence: The paper section correspondence identifies the section of the paper on which the review statement is commenting. E.g, Abstract (ABS), Introduction (INT), Related Works (RWK), Problem Definition/Idea (PDI), Data/Datasets (DAT), Methodology (MET), Experiments (EXP), Results (RES), Tables & Figures (TNF), Analysis (ANA), Future Work (FWK),

[2]https://peerreviewcongress.org/

[3]https://iclr.cc/Conferences/2022/MetareviewGuide

[4]https://acl2020.org/reviewers/

[5]https://neurips.cc/Conferences/2022/ReviewerGuidelines

[6]https://icml.cc/Conferences/2022/ReviewerTutorial

Dataset	# Purpose	# Review	Avg. length of review (terms of words)	Avg. length of review (terms of sentences)	# Annotated sentences
ICLR 2018	For proposed	1322	345.878	17.511	23150

Table 1: Dataset statistics

Overall (OAL), Bibliography (BIB) and External (EXT).

Paper Aspect Category: The paper aspect category identifies the aspect of the paper that the review-statement addresses. E.g, Appropriateness (APR), Originality or Novelty (NOV), Significance or Impact (IMP), Meaningful Comparison (CMP), Presentation & Formatting (PNF), Recommendation (REC), Empirical & Theoretical Soundness (EMP), Substance (SUB) and Clarity (CLA).

Reviewer - Centric Uncertainty: In peer review, reviewers sometimes make superficial, speculative comments, which are not very helpful, and ultimately affect the outcome (Ghosal et al., 2022b; Özgür and Radev, 2009). For example, some reviewers use vague or hedge words (e.g., maybe, seems, might, etc.) when uncertain about their review. There could be discrepancies between how reviewers comment on themselves and how readers see their preview text. This intuition suggests that a good review will have less reviewer-centric uncertainty (low hedge score). Therefore, we incorporate reviewer-centric uncertainty into our proposed method.

Informativeness Score: Reviews that cover the complete work are more likely helpful to the author (Kühne et al., 2010). It can be an indication of how detailed and significant the judgment was with this intuition. We identify the study corresponding to the paper section and aspects within reviews. The main idea is to arrive at a justifiable informativeness score; if a review is good, it will cover as many sections and important aspects as possible. With this objective, we encoded the annotation label into a numerical score based on the review's coverage across section correspondence, aspect category and reviewer-centric uncertainty of the review by measuring the informativeness score towards the automatic judgment of review quality. We have calculated the informativeness score by considering following three parameters.

3.1.1 i) Section Score ($R_{\textbf{sec}}$):

A good review should comment on the important sections of the paper, which may help us identify whether the reviewer's comments are semantically related to the submission's main contents. With this intuition, we calculate the section score by given formula.

$$R_{sec} = \frac{\sum \bar{x}_i + \sum \mu_i W_{xi}}{\sum x_i} \quad (1)$$

Where $\Sigma\bar{x}_i$ = no. of unique sections covered by review, μ_i = no. of repeating sentences containing i$^\text{th}$ section, W_xi = weight of i$^\text{th}$ section and $\sum x_i$ = total no. of sections.

3.1.2 ii) Aspect Score ($R_{\textbf{asp}}$):

As per the rubrics defined (Yuan et al., 2021; Ghosal et al., 2022a) in Peer Review Analyze paper, we expect the review to evaluate the work for indicators like novelty, theoretical and empirical soundness of the research methodology, writing, and clarity of the work, impact of the work in a broader academic context, etc. We call these indicators review-level aspects. We calculate aspect score using the following formula.

$$R_{\text{asp}} = \frac{\sum \bar{x}_i + \sum \mu_i w_{\text{xi}}}{\sum x_i} \quad (2)$$

Where $\Sigma\bar{x}_i$ = no. of unique aspects covered, μ_i = no. of repeating sentences containing i$^\text{th}$ aspect, w_xi = weight of i$^\text{th}$ aspect $\sum x_i$ = total no. of aspects.

3.1.3 Assigning the Weights W$_{xi}$:

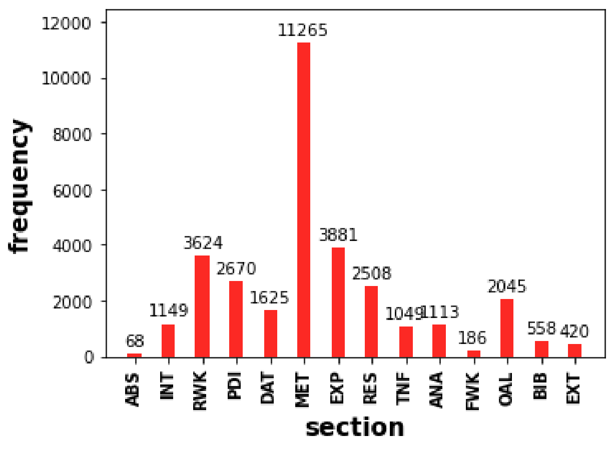
Figure 1 shows the label distribution for each review across the datset for sections and aspects layer. And we assign the weight to respective sections and aspects in our informativeness formula accordingly.

$$W_{xi} = \frac{\text{Freq}_{\text{ xi}}}{100 * \text{Total Freq}} \quad (3)$$

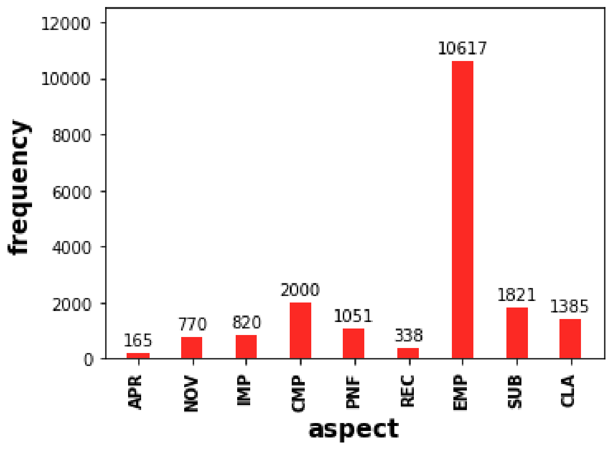
Freq $_\text{xi}$ = number of sentences talking about a specific section/aspect, Total freq: total number of sentences talking about sections/aspects.

3.1.4 iii) Reviewer-Centric Uncertainty (Hedge Score (H)):

In a review, uncertainty refers to speculation made by the reviewer. The words the reviewer uses to indicate speculating are called hedge words (Lakoff,



(a) Sections distribution



(b) Aspect distribution

Figure 1: Sections and aspects distribution across paper section correspondence and paper aspect category in Peer Review Analyze annotated dataset.

1970; Tang et al., 2010; Velldal et al., 2012). Counting uncertain terms in a review is normalized with the number of words in a review to calculate hedge scores. To calculate the hedge score, we use the method proposed by Khandelwal A. et al. (Britto and Khandelwal, 2020; Khandelwal and Sawant, 2019), and we use the XLNet (Yang et al., 2019) version since it outperforms BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019).

$$\text{Hedge Score} = \frac{\Sigma(\text{ hedge words })}{\Sigma(\text{ words })} \tag{4}$$

The score ranges from 0 to 1. If a reviewer is uncertain the hedge score will be higher and vice versa.
Based on the above discussion and using Equations 1, 2, 3 and 4, we derive an informativeness score for a review, which is given below.

$$\textbf{Informativeness score}(R_{\text{info}}) = \frac{R_{\text{sec}}}{e^{H} * e^{1-R_{\text{asp}}}} \tag{5}$$

Where R_{info} = Informativeness score, R_{sec} = Section score, R_{asp} = Aspect score and H = Hedge score.

3.1.5 Intuition about the informativeness score:

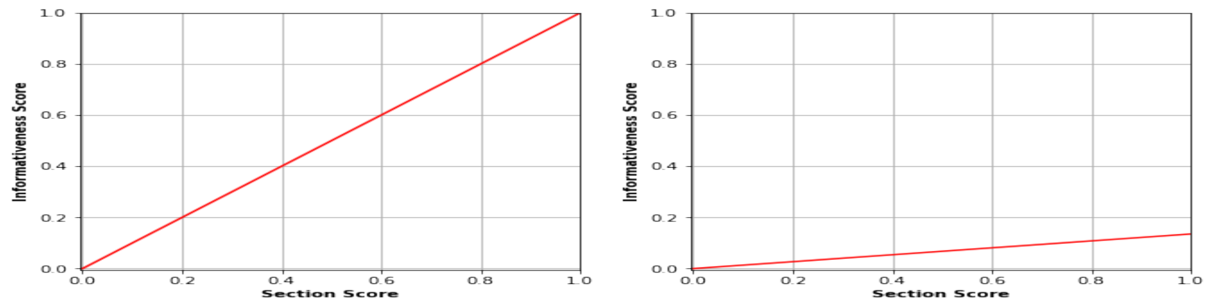
We plot the graph between the informativeness score (R_{info}) and the other three parameters (in the best and worst case). We consider this observation in the informativeness score formula accordingly.

Section Score (R_{sec})**:** From Figure 2, we can see the reason to keep the section score in the numerator.

- Informativeness score is directly proportional to section score $R_{\text{info}} \propto R_{\text{sec}}$ and hence, higher the R_{sec}, higher will be the R_{info} .
- The section score has the highest contribution in determining the informativeness score; as when section score $= 0$, irrespective of the other two parameters, informativeness score will always be $= 0$ (see Figure 2.)

Aspect Score (R_{asp})**:** Figure 3 illustrates the relation between informativeness score and aspect score.

- From Figure 3, we can see that higher the aspect score, lower is the $(1 - R_{\text{asp}})$, and hence and value of e^ $(1 - R_{\text{asp}})$ is lower, higher will be the informativeness score. Aspect score has a lower contribution to the informativeness score, as even when aspect score = 0, informativeness score still can be upto 0.3679, depending on the other two parameters (section and hedge score).
- We intend that the informativeness score increases exponentially with increasing aspect score hence, $R_{\text{info}} \propto e^{\wedge}R_{\text{asp}}$. However, to limit the max. R_{info} to 1 at $R_{\text{asp}} = 1$ (Best condition when section score = 1, hedge score = 0) and max. aspect score = 1, we divide the informativeness score by a factor of e.



(a) Best condition (when aspect score = 1 and hedge score = 0) (b) Worst condition (when aspect score = 0 and hedge score = 1)

Figure 2: Informativeness Score Vs. Section Score.

Therefore, $\mathrm{R_{info}} \propto (\mathrm{e}^\wedge \mathrm{R_{asp}})/\mathrm{e}$, which implies that $\mathrm{R_{info}} \propto \mathrm{e}^\wedge(\mathrm{R_{asp}} - 1)$. Hence $\mathrm{R_{info}} \propto 1/\mathrm{e}^\wedge(\mathbf{1} - \mathrm{R_{asp}})$.

Hedge Score (H)**:** Figure 4 illustrates the reason to keep hedge score in the denominator, as a power of e, such that $R_{\text{info}} \propto 1/e^\wedge\mathrm{H}$.

- So, higher the hedge score, higher the $\mathrm{e}^\wedge\mathrm{H}$, and hence lower will be the informativeness score.
- we can see from Figure 4 hedge score has a lower contribution to the informativeness score; as even when hedge score = 1, informativeness score can reach 0.3679, depending on the other two parameters (section and aspect score).
- We intend that the informativeness score decreases exponentially with increasing hedge score, and at $\mathrm{H} = 0$, $\mathrm{R_{info}} = 1$. Hence, $\mathrm{R_{info}} \propto \mathrm{e}^\wedge(-\mathrm{H})$ which implies that $\mathrm{R_{info}} \propto 1/\mathrm{e}^\wedge\mathrm{H}$.

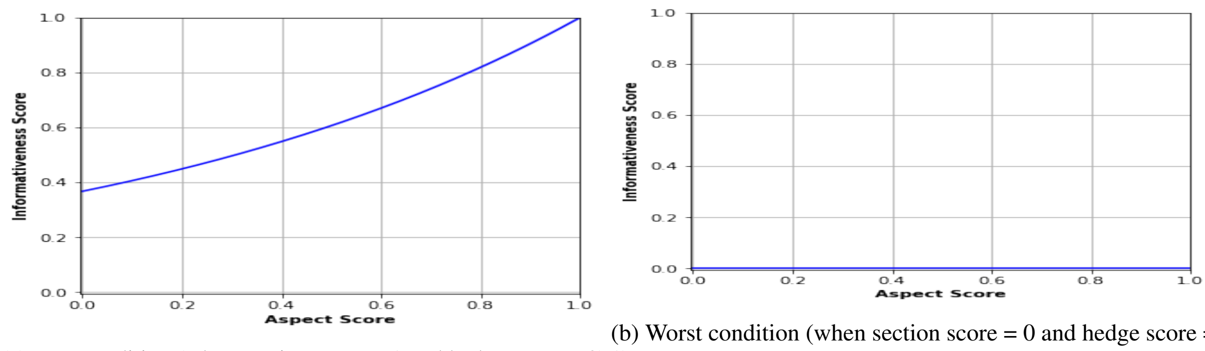
4 Benchmarking Experiments

In addition, we provide baselines for natural language processing (NLP) on the experimental dataset (both annotated and unannotated). Moreover, we train nine methods based on data, including Multiple Linear Regressions (MLR), Robust Regressions (RANSAC), Random Forest Regressions (RF), Long Short-Term Memory (LSTM), Extreme Learning Machines (ELM), Bidirectional Long Short-Term Memory (BiLSTM), Masked and Permuted Pre-training for Language Understanding (MPNet) (Song et al., 2020), Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018), as well as Transformer variants of SciBERT (Beltagy et al., 2019).

4.1 Features for Peer Review Analyze (annotated dataset)

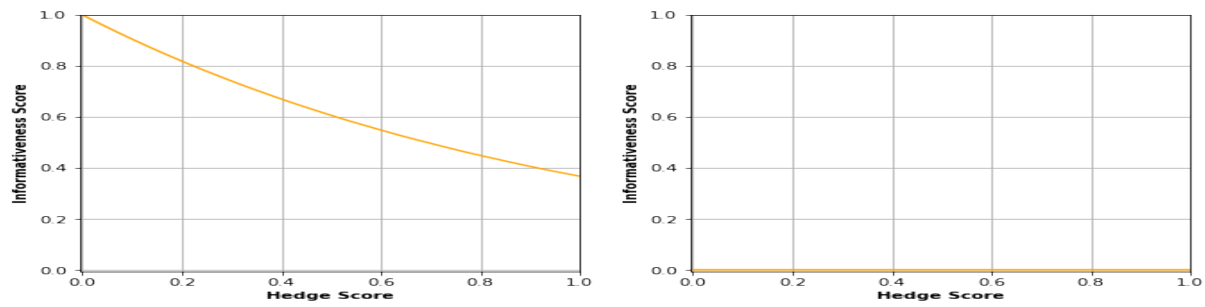
We use a set of features that includes:

- **Sentence and word count :** We have used the five features sentence count, word count, average sentence length, average word length, and vocab length. The informativeness score is directly proportional to the length of review sentence count and word count, as well as the size of vocabulary vocab length. This gives us a feature matrix of dimension 5.
- **Hedge features:** For review uncertainty, we use the hedge feature hedgescore, which is the average hedge words per sentence, where the hedge words are determined by the method proposed by Khandelwal A. et al. (Britto and Khandelwal, 2020; Khandelwal and Sawant, 2019). This gives us a feature matrix of dimension 1.
- **PoS features:** PoS (Parts of Speech) includes nouns, adjectives, verbs, and adverbs.
- **Sentiment features:** We use VADER (Valence Aware Dictionary for Sentiment Reasoning) (Hutto and Gilbert, 2014) compound sentiment score as the sentiment feature. It ranges from -1 to 1 and gives a feature matrix of dimension 1.
- **Keyword count:** We take the 50 most appearing terms from the papers with top 20% informativeness score as keywords, hence obtaining a feature matrix of 50.
- **Section and aspect coverage:** We use the number of sections covered (out of 14) and the number of aspects covered (out of 9), by



(a) Best condition (when section score = 1 and hedge score = 0) (b) Worst condition (when section score = 0 and hedge score = 1)

Figure 3: Informativeness Score Vs. Aspect Score



(a) Best condition (when aspect score = 1 and aspect score = 1) (b) Worst condition (when aspect score = 0 and aspect score = 0)

Figure 4: Informativeness Score Vs. Hedge Score.

the review as features, with feature matrix dimension 2.

- **Section and aspect distribution:** We take the counts of the number of sentences in the review that talks about each section/aspect as features. This gives us a feature of dimension 23.

4.2 Features for unannotated dataset

We use a set of features, which includes sentence and word counts, sentiment features, PoS (Part of Speech), i.e., nouns, adjectives, verbs, and adverbs, hedge features, and keyword counts. Kindly refer to our GitHub repository for the definition and implementation of our full feature set.
Thus, we use feature matrices of dimension 86 for annotated reviews and dimension 61 for unannotated review text (for both, we use Peer Review Analyze dataset) to predict informativeness scores. In addition, word embeddings of their specific dimensions to deep learning models with the Bidirectional Long Short-Term Memory (BiLSTM) pipeline, we use a standard implementation of machine learning models from sci-kit python library, (Pedregosa et al., 2011) keeping the default parameters fixed for a fair comparison across variations in models and embeddings.
Implementation Details: We use Keras on top of TensorFlow-2.4.1 to build the model. Moreover, we train the model with batch size 32, and Adam optimizer with a weight_decay = $\{1e{-}3\}$ to avoid overfitting, and kept each batch balanced while training. We use fixed set $\{1e-1, 1e-2, 1e-3, 3e-3\}$ to tune the learning rate, and find $\{1e-3\}$ works best in our experimental setup. Please see our repository link in the abstract for further information.

4.3 Experimental Setup

In terms of our experimental setup, we use more than one evaluation metrics to avoid any confusion. Because different metrics with the same data can produce different values. It is always better to have a combination of metrics-like MAE (Mean absolute error), Root mean square error (RMSE) and coefficient of determination (R^2) to use together and apply the same metric on a different model to see which one produces the best performance.

5 Evaluation Results & Analysis

We report the evaluation results for annotated and unannotated datasets in Table 2 and Table 3. We kept 80% of the data for training and 20% for eval-

Model Types	MAE	RMSE	(R^2)
MLR	0.0205	0.0305	0.9061
RANSAC	0.0201	0.0295	0.9167
RF	0.0181	0.1924	0.9297
LSTM	0.0178	0.0286	0.9331
ELM	0.0171	0.0267	0.9435
BiLSTM	0.0191	0.0219	0.9619
MPNet	0.0162	0.0184	0.9730
BERT	0.0197	0.0229	0.9583
SciBERT	**0.0152**	**0.0171**	**0.9871**

Table 2: Performance comparision for qualitative analysis on annotated dataset in terms of MAE, RMSE and R-squared (R^2).

Model Types	MAE	RMSE	(R^2)
MLR	0.0596	0.0787	0.3212
RANSAC	0.0666	0.0864	0.3276
RF	0.0682	0.0894	0.3656
LSTM	0.0646	0.0935	0.3051
ELM	0.0636	0.0810	0.3787
BiLSTM	0.0659	0.0878	0.3875
MPNet	0.0657	0.0954	0.2767
BERT	0.0711	0.0931	0.3115
SciBERT	**0.0621**	**0.0735**	**0.4155**

Table 3: Performance comparison for qualitative analysis on unannotated dataset in terms of MAE, RMSE and R-squared (R^2).

uation of the models. We experiment with nine data-driven methods: Multiple Linear Regression (MLR), Robust Regression (RANSAC), Random Forest Regression (RF), Long Short-Term Memory (LSTM), Extreme Learning Machines (ELM), Bidirectional Long Short-Term Memory (BiLSTM), Masked and Permuted Pre-training for Language Understanding (MPNet), Bidirectional Long-Short Term Memory (BiLSTM) on Bidirectional Encoder Representations from Transformers (BERT), and a Bidirectional Long-Short Term Memory (BiLSTM) on Transformer variant of SciBERT, to test the proposed proposition. As shown in Table 2 and Table 3, the deep neural model based on SciBERT representations outperforms both annotated and unannotated datasets.

Qualitative Analysis on Baseline Models: Table 4 shows informativeness score calculate by proposed method and automatically generated informativeness score by nine different techniques on a given Neural Information Processing Systems (NeurIPS) reviews. For qualitative analysis, we take our trained models and predict the score on Neural Information Processing Systems (NeurIPS) sample reviews dataset from the open-access platform OpenReview platform[7]. Table 4 shows some examples of them.

5.1 Case Study:

We analyzed the two ICLR reviews qualitatively to support our proposed method. In the review $https://openreview.net/forum?id=B1EA-M-0Z$. We can see that out of 14 sections, the review has covered 8 unique sections, out of 9 aspects, it covers 4 unique aspects, and this review also has a reviewer-centric uncertainty calculated by hedge score. We can see from Figure 5 (a) the following observations.

- If the review has higher coverage in sections and aspects, the higher will be the section and aspect score. It leads to a higher informativeness score.
- If the reviewer-centric uncertainty (hedge score) is high, then informativeness should be low.

$https://openreview.net/forum?id=ByuP8yZRb$, we can see that out of 14 sections, the review has covered only 6 unique sections, and out of 9 aspects, it covers 3 unique aspects, and this review has high reviewer-centric uncertainty calculated by hedge score. The following observations can be seen in Figure 5 (b).

- This review has low coverage in terms of sections and aspects. Due to this, it has a low informativeness score.
- This review has a high reviewer-centric uncertainty in terms of hedge score, leading to a low informativeness score.

In summary, from this case study shown in Figure 5, we can see the efficiency and suitability of the proposed informativeness method.

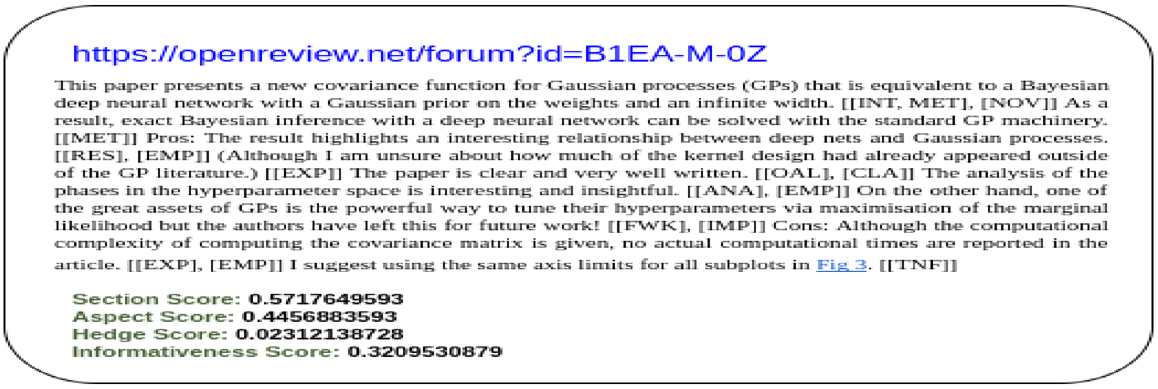
6 Conclusion and future work

In this paper, we provide an effective solution to automatically estimate the informativeness score

[7]https://openreview.net/

Review Id	(Informativeness score calculate by proposed method)	Informativeness Score Predicted by Baseline Models								
		MLR	RANSA	RF	LSTM	BiLSTM	ELM	MPNet	BERT	SciBERT
URL: https://proceedings.neurips.cc/paper/2018/file/9246444d94f081e3549803b928260f56-Reviews.html										
NIPS_2018_1006__R1	0.1596	0.1108	0.1176	0.1292	0.1316	0.1381	0.1328	0.1398	0.1347	0.1443
NIPS_2018_1006__R2	0.2713	0.1849	0.1989	0.1998	0.2189	0.2191	0.2212	0.2351	0.2479	0.2569
NIPS_2018_1006__R3	0.5053	0.3992	0.4087	0.4097	0.4162	0.4194	0.4276	0.4459	0.4639	0.4752
URL: https://proceedings.neurips.cc/paper/2018/file/e77dbaf6759253c7c6d0efc5690369c7-Reviews.html										
NIPS_2018_443__R1	0.2822	0.1818	0.1884	0.1931	0.2245	0.2279	0.2311	0.2434	0.2496	0.2765
NIPS_2018_443__R2	0.3249	0.2067	0.2107	0.2256	0.2383	0.2338	0.2430	0.2458	0.2961	0.3006
NIPS_2018_443__R3	0.3236	0.2022	0.2308	0.2355	0.2412	0.2443	0.2536	0.2563	0.3038	0.3151

Table 4: Qualitative analysis results for predicting the Informativeness score by baseline models.

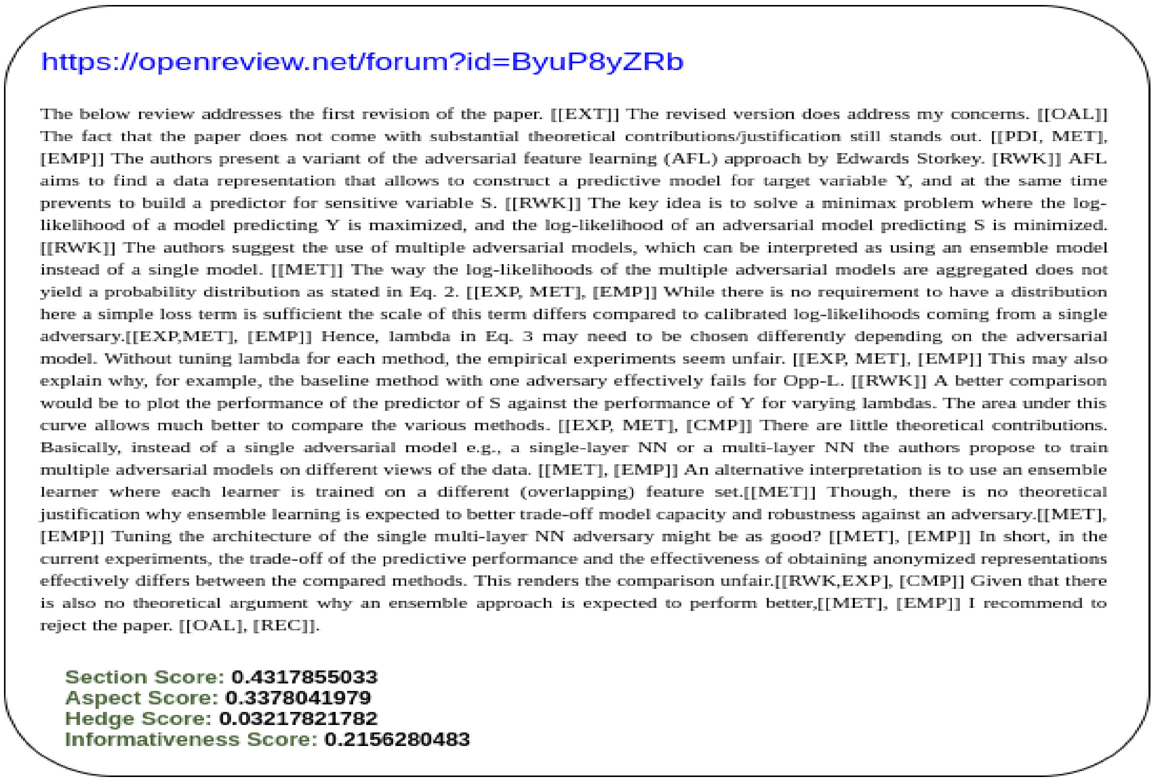


https://openreview.net/forum?id=B1EA-M-0Z

This paper presents a new covariance function for Gaussian processes (GPs) that is equivalent to a Bayesian deep neural network with a Gaussian prior on the weights and an infinite width. [[INT, MET], [NOV]] As a result, exact Bayesian inference with a deep neural network can be solved with the standard GP machinery. [[MET]] Pros: The result highlights an interesting relationship between deep nets and Gaussian processes. [[RES], [EMP]] (Although I am unsure about how much of the kernel design had already appeared outside of the GP literature.) [[EXP]] The paper is clear and very well written. [[OAL], [CLA]] The analysis of the phases in the hyperparameter space is interesting and insightful. [[ANA], [EMP]] On the other hand, one of the great assets of GPs is the powerful way to tune their hyperparameters via maximisation of the marginal likelihood but the authors have left this for future work! [[FWK], [IMP]] Cons: Although the computational complexity of computing the covariance matrix is given, no actual computational times are reported in the article. [[EXP], [EMP]] I suggest using the same axis limits for all subplots in Fig 3. [[TNF]]

Section Score: 0.5717649593
Aspect Score: 0.4456883593
Hedge Score: 0.02312138728
Informativeness Score: 0.3209530879

(a) Informativeness score calculated by proposed method



https://openreview.net/forum?id=ByuP8yZRb

The below review addresses the first revision of the paper. [[EXT]] The revised version does address my concerns. [[OAL]] The fact that the paper does not come with substantial theoretical contributions/justification still stands out. [[PDI, MET], [EMP]] The authors present a variant of the adversarial feature learning (AFL) approach by Edwards Storkey. [RWK]] AFL aims to find a data representation that allows to construct a predictive model for target variable Y, and at the same time prevents to build a predictor for sensitive variable S. [[RWK]] The key idea is to solve a minimax problem where the log-likelihood of a model predicting Y is maximized, and the log-likelihood of an adversarial model predicting S is minimized. [[RWK]] The authors suggest the use of multiple adversarial models, which can be interpreted as using an ensemble model instead of a single model. [[MET]] The way the log-likelihoods of the multiple adversarial models are aggregated does not yield a probability distribution as stated in Eq. 2. [[EXP, MET], [EMP]] While there is no requirement to have a distribution here a simple loss term is sufficient the scale of this term differs compared to calibrated log-likelihoods coming from a single adversary.[[EXP,MET], [EMP]] Hence, lambda in Eq. 3 may need to be chosen differently depending on the adversarial model. Without tuning lambda for each method, the empirical experiments seem unfair. [[EXP, MET], [EMP]] This may also explain why, for example, the baseline method with one adversary effectively fails for Opp-L. [[RWK]] A better comparison would be to plot the performance of the predictor of S against the performance of Y for varying lambdas. The area under this curve allows much better to compare the various methods. [[EXP, MET], [CMP]] There are little theoretical contributions. Basically, instead of a single adversarial model e.g., a single-layer NN or a multi-layer NN the authors propose to train multiple adversarial models on different views of the data. [[MET], [EMP]] An alternative interpretation is to use an ensemble learner where each learner is trained on a different (overlapping) feature set.[[MET]] Though, there is no theoretical justification why ensemble learning is expected to better trade-off model capacity and robustness against an adversary.[[MET], [EMP]] Tuning the architecture of the single multi-layer NN adversary might be as good? [[MET], [EMP]] In short, in the current experiments, the trade-off of the predictive performance and the effectiveness of obtaining anonymized representations effectively differs between the compared methods. This renders the comparison unfair.[[RWK,EXP], [CMP]] Given that there is also no theoretical argument why an ensemble approach is expected to perform better,[[MET], [EMP]] I recommend to reject the paper. [[OAL], [REC]].

Section Score: 0.4317855033
Aspect Score: 0.3378041979
Hedge Score: 0.03217821782
Informativeness Score: 0.2156280483

(b) Informativeness score calculated by proposed method

Figure 5: Qualitative analysis on annotated ICLR Reviews.

of review on the shoulder of uncertainty and review coverage (sections and aspects of the paper). For the proposed method, we used a publicly available Peer Review Analyze dataset of peer review texts, manually annotated at the sentence level (13.22k sentences) across two layers: Paper Section Correspondence and Paper Aspect Category. Next, we transform these categorical annotations to derive an informativeness score of the review based on the review's coverage across section correspondence, aspects of the paper, and reviewer-centric uncertainty associated with the review toward the automatic judgment of review quality. We believe that these interpretations can assist the editors in making better editorial decisions.

References

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*.

Prabhat Kumar Bharti, Tirthankar Ghosal, Mayank Agrawal, and Asif Ekbal. 2022a. How confident was your reviewer? estimating reviewer confidence from peer review texts. In *International Workshop on Document Analysis Systems*, pages 126–139. Springer.

Prabhat Kumar Bharti, Asheesh Kumar, Tirthankar Ghosal, Mayank Agrawal, and Asif Ekbal. 2022b. Can a machine generate a meta-review? how far are we? In *International Conference on Text, Speech, and Dialogue*, pages 275–287. Springer.

Prabhat Kumar Bharti, Shashi Ranjan, Tirthankar Ghosal, Mayank Agrawal, and Asif Ekbal. 2021. Peerassist: Leveraging on paper-review interactions to predict peer review decisions. In *International Conference on Asian Digital Libraries*, pages 421–435. Springer.

Elise S Brezis and Aliaksandr Birukou. 2020. Arbitrariness in the peer review process. *Scientometrics*, 123(1):393–411.

Benita Kathleen Britto and Aditya Khandelwal. 2020. Resolving the scope of speculation and negation using transformer-based architectures. *arXiv preprint arXiv:2001.02885*.

Rachel Bruce, Anthony Chauvin, Ludovic Trinquart, Philippe Ravaud, and Isabelle Boutron. 2016. Impact of interventions to improve the quality of peer review of biomedical journals: a systematic review and meta-analysis. *BMC medicine*, 14(1):1–16.

Michael L Callaham, William G Baxt, Joseph F Waeckerle, and Robert L Wears. 1998. Reliability of editors' subjective quality ratings of peer reviews of manuscripts. *Jama*, 280(3):229–231.

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*.

Rafael D'Andrea and James P O'Dwyer. 2017. Can editors save peer review from peer reviewers? *PloS one*, 12(10):e0186111.

Martin Enserink. 2001. Peer review and quality: A dubious connection?

Yang Gao, Steffen Eger, Ilia Kuznetsov, Iryna Gurevych, and Yusuke Miyao. 2019. Does my rebuttal matter? insights from a major nlp conference. *arXiv preprint arXiv:1903.11367*.

Tirthankar Ghosal. 2019. Exploring the implications of artificial intelligence in various aspects of scholarly peer review. *Bull. IEEE Tech. Comm. Digit. Libr*, 15(1).

Tirthankar Ghosal, Sandeep Kumar, Prabhat Kumar Bharti, and Asif Ekbal. 2022a. Peer review analyze: A novel benchmark resource for computational analysis of peer reviews. *Plos one*, 17(1):e0259238.

Tirthankar Ghosal, Kamal Kaushik Varanasi, and Valia Kordoni. 2022b. Hedgepeer: a dataset for uncertainty detection in peer reviews. In *Proceedings of the 22nd ACM/IEEE Joint Conference on Digital Libraries*, pages 1–5.

Tirthankar Ghosal, Rajeev Verma, Asif Ekbal, and Pushpak Bhattacharyya. 2019. Deepsentipeer: Harnessing sentiment in review texts to recommend peer review decisions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1120–1130.

Ann T Gregory and A Robert Denniss. 2019. Everything you need to know about peer review—the good, the bad and the ugly. *Heart, Lung and Circulation*, 28(8):1148–1153.

Clayton Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 8.

Tom Jefferson, Elizabeth Wager, and Frank Davidoff. 2002. Measuring the quality of editorial peer review. *Jama*, 287(21):2786–2790.

Amy C Justice, Mildred K Cho, Margaret A Winker, Jesse A Berlin, Drummond Rennie, Peer Investigators, PEER Investigators, et al. 1998. Does masking author identity improve peer review quality?: A randomized controlled trial. *Jama*, 280(3):240–242.

Aditya Khandelwal and Suraj Sawant. 2019. Negbert: a transfer learning approach for negation detection and scope resolution. *arXiv preprint arXiv:1911.04211*.

Conny Kühne, Klemens Böhm, and Jing Zhi Yue. 2010. Reviewing the reviewers: A study of author perception on peer reviews in computer science. In *6th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2010)*, pages 1–8. IEEE.

Asheesh Kumar, Tirthankar Ghosal, and Asif Ekbal. 2021. A deep neural architecture for decision-aware meta-review generation. In *2021 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 222–225. IEEE.

Sandeep Kumar, Hardik Arora, Tirthankar Ghosal, and Asif Ekbal. 2022. Deepaspeer: towards an aspect-level sentiment controllable framework for decision prediction from academic peer reviews. In *Proceedings of the 22nd ACM/IEEE Joint Conference on Digital Libraries*, pages 1–11.

George Lakoff. 1970. Linguistics and natural logic. *Synthese*, 22(1):151–271.

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*.

Arzucan Özgür and Dragomir Radev. 2009. Detecting speculations and their scopes in scientific text. In *Proceedings of the 2009 conference on empirical methods in natural language processing*, pages 1398–1407.

F Pedregosa, G Varoquaux, A Gramfort, V Michel, B Thirion, and O Grisel. 2011. Blonde, l m. *Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay E*, pages 2825–2830.

Drummond Rennie. 2016. Let's make peer review scientific. *Nature*, 535(7610):31–33.

Mona M Shattell, Peggy Chinn, Sandra P Thomas, and W Richard Cowling III. 2010. Authors' and editors' perspectives on peer review quality in three scholarly nursing journals. *Journal of nursing scholarship*, 42(1):58–65.

Kyle Siler, Kirby Lee, and Lisa Bero. 2015. Measuring the effectiveness of scientific gatekeeping. *Proceedings of the National Academy of Sciences*, 112(2):360–365.

Amanda Sizo, Adriano Lino, Luis Paulo Reis, and Álvaro Rocha. 2019. An overview of assessing the quality of peer review reports of scientific articles. *International Journal of Information Management*, 46:286–293.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. *Advances in Neural Information Processing Systems*, 33:16857–16867.

Buzhou Tang, Xiaolong Wang, Xuan Wang, Bo Yuan, and Shixi Fan. 2010. A cascade method for detecting hedges and their scope in natural language text. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning–Shared Task*, pages 13–17.

Susan Van Rooyen. 2001. The evaluation of peer-review quality. *Learned Publishing*, 14(2):85–91.

Susan Van Rooyen, Nick Black, and Fiona Godlee. 1999. Development of the review quality instrument (rqi) for assessing peer reviews of manuscripts. *Journal of clinical epidemiology*, 52(7):625–629.

Erik Velldal, Lilja Øvrelid, Jonathon Read, and Stephan Oepen. 2012. Speculation and negation: Rules, rankers, and the role of syntax. *Computational linguistics*, 38(2):369–410.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Weizhe Yuan, Pengfei Liu, and Graham Neubig. 2021. Can we automate scientific reviewing? *arXiv preprint arXiv:2102.00176*.

Spellchecker for Sanskrit: The Road Less Taken

Prasanna Venkatesh T S
Ph.D. Research Scholar
Department of Sanskrit
Ramakrishna Mission
Vivekananda College (Autonomous),
Chennai - 600004.
`vipranarayan14@gmail.com`

Abstract

A spellchecker is essential for any language for producing error-free content. While there exist advanced computational tools for Sanskrit, such as word segmenter, morphological analyser, sentential parser, and machine translation, a fully functional spellchecker is not available. This paper presents a Sanskrit spellchecking dictionary for Hunspell, thereby creating a spellchecker that works across the numerous platforms Hunspell supports. The spellchecking rules are created based on the Paninian grammar, and the dictionary design follows the word-and-paradigm model, thus, making it easily extendible for future improvements. The paper also presents an online spellchecking interface for Sanskrit developed mainly for the platforms where Hunspell integration is not available yet.

1 Introduction

A spellchecker is a program that checks for misspellings in a text and suggests correct alternatives (Lawaye and Purkayastha, 2016). It is an essential tool for word processing and document preparation.

In the recent decade, digitization of Sanskrit manuscripts and using digital technologies for editing and publishing new content in Sanskrit have seen a tremendous increase. Online Sanskrit communities and many individuals are actively using Sanskrit in writing blogs, emails and chat messages, while some are designing posters in Sanskrit. A spellchecker can help them to communicate clearly and produce error-free content.

After the digitization of a manuscript through an OCR (optical character recognition), often there might be errors in the text due to the visual similarity in certain letters such as ब-व and य-थ (Schnober et al., 2016). These can be corrected using a spellchecker manually or through a pipeline that performs the OCR post-correction automatically. Also, advanced tools like morphological analyser (Kulkarni and Shukl, 2009; Huet, 2005) and sentential parser (Kulkarni, 2019) might produce incorrect results or no results at all, if the input contains spelling errors (Murthy et al., 2012). In such cases, the input can be pipelined through a spellchecker for pre-checking and pre-processing.

A spellchecker can also sometimes help new learners of Sanskrit in learning the correct spellings of words, especially, of those having phonetically similar letters such as श-ष in "शेष", ए-ये in "एक", न-ण in "बाणेन", etc.

This paper presents a Sanskrit spellchecking dictionary for Hunspell based on the word-and-paradigm model and describes its design and implementation. It discusses the suitability of Hunspell for a highly inflectional language like Sanskrit (Section 3) and the format of a Hunspell dictionary (Section 4). It also describes the framing of spellchecking rules based on Paninian grammar (Section 5). The paper also presents a web interface for Sanskrit spellchecking (Section 6). Later, it discusses the evaluation of both the spellchecking dictionary and the web interface (Section 7). The paper also records the challenges unique to Sanskrit in creating a spellchecking dictionary (section 8).

2 Related Works and Previous Attempts

While there exist advanced computational tools for Sanskrit such as word segmenter (Hellwig and Nehrdich, 2018), morphological analyser (Kulkarni and Shukl, 2009), sentential parser (Kulkarni, 2019), machine translation (Kulkarni, 2009), automatic speech recognition (Adiga et al., 2021), etc., a fully functional spellchecker is not available. Spellcheckers and grammar checkers are fundamental tools that users need and expect nowadays with the increase in the use of digital technologies for both creating new content as well as digitising existing texts.

Significant work has been done on spellcheckers for other Indian languages like Hindi (Kaur and Singh, 2015; Pathan et al., 2019; Kanwar et al., 2017; Jain et al., 2018), Marathi (Dixit et al., 2016), Odia (Pradhan and Dalai, 2020), Punjabi (Lehal, 2007), Kashmiri (Lawaye and Purkayastha, 2016), Telugu (Uma Maheshwar Rao G. et al., 2012), Tamil (Segar and Kengatharaiyer, 2015) and Kannada (Murthy et al., 2012, 2017).

Though there is not much research work available on spellchecker for Sanskrit, there were a few attempts in the past to develop one. Tapaswi et al. (2012) proposed and developed a spellchecker based on the morphological rules of Sanskrit. It was a standalone application implemented in Java. Samsādhanī[1] developed and hosted a spellchecker web application where their morphological analyser runs on the text provided by a user and highlights incorrect words. The application, however, is no longer maintained[2]. Patel (2016) built a Sanskrit spellchecker that works based on different vowel and consonant patterns. But it was specific to the Cologne Sanskrit dictionaries, and not for general spellchecking (Patel, 2021). Gasuns (2013) and Kumar (2017) independently created Sanskrit dictionaries for Hunspell. But, as of now, both the dictionaries are not complete and are not maintained. Quintanilha and Líbera (2018) developed a dictionary add-on for Mozilla Firefox which contains a Sanskrit Hunspell dictionary. However, it is also far from complete.

Other than these, to the best of our knowledge, there has been no significant progress in the development of a Sanskrit spellchecking dictionary for Hunspell or any other spellchecker for Sanskrit.

[1]`https://scl.samsaadhanii.in/scl/`

[2]Dr. Amba Kulkarni, personal communication.

3 Why Hunspell?

Hunspell is a free open-source spellchecker and morphological analyser library and also a command-line tool[3]. It is the most popular spellchecker that is used in many applications like LibreOffice[4], Mozilla Firefox[5], Google Chrome[6] and Adobe InDesign[7] and has bindings in numerous popular programming languages (Németh, 2019). In Linux and macOS, after installing the dictionaries the users can enjoy system-wide spellchecking. Such tight integration helps correct misspellings even as the user is writing instead of having them copy and paste the text into an external program just for spellchecking. Therefore, designing a dictionary for Hunspell means we can have Sanskrit spellchecking on almost all the platforms.

Sanskrit is a highly inflectional language and highly productive in derivative morphology such as *Samāsa*, *Kṛdanta* and *Taddhita* (Adiga et al., 2018). A simple list of all the correct forms would be extremely huge in terms of computer storage. Hunspell format allows us to define the base forms and affixes separately which hugely reduces the dictionary size. This division also makes it easier to add new words to the dictionary.

4 Format of Hunspell Dictionary Files

Before going into the design and preparation of the Hunspell dictionary, the format of the dictionary files is briefly discussed here. Hunspell requires two files to define how it should spellcheck for a language and suggest correct alternatives[8] – **a dictionary file** and **an affix file**. The first line of the dictionary file contains the (approximate) count of the number of entries in the file (Németh, 2018). From

[3]`https://github.com/hunspell/hunspell/`

[4]`https://extensions.libreoffice.org/?q=dictionary`

[5]`https://addons.mozilla.org/en-US/firefox/language-tools/`

[6]`https://chromium.googlesource.com/chromium/deps/hunspell_dictionaries/`

[7]`https://helpx.adobe.com/indesign/kb/add_cs_dictionaries.html`

[8]The dictionary and affix file format is discussed here only briefly to understand the design of the current dictionary. For more information refer (Németh, 2018) and (Shepelev, 2021).

sa_IN.dic	**sa_IN.aff**
37058	SFX 1001 Y 17
...	SFX 1001 ◌िस् ◌ीः .
आशिस्/1001	SFX 1001 स् षौ .
अर्च्चिस्/1001	SFX 1001 स् षः .
भुविस्/1001	SFX 1001 स् षम् .
...	...

Table 1: Samples from the final **dictionary** (sa_IN.dic) and **affix** (sa_IN.aff) files.

the second line onwards, we have the entries, one per line. An entry can be a morpheme or lexeme or can even be a pair of words, and it can be optionally followed by a forward slash ("/") and one or more "flags" which represent its attributes such as suffix, prefix, etc. Table 1 (sa_IN.dic) is a sample from the final dictionary file showing the entries' count, some entries and their flags.

The definitions of these flags are given in the **affix file**. Table 1 (sa_IN.aff) is a sample from the affix file showing the **affix class** corresponding to the flag "1001", used in the dictionary. An affix class definition consists of a **header** (the first line) followed by a number of **affix rules**, each separated by a new line. The affix header consists of four fields describing:

- Whether the class is for a suffix or prefix.
- The paradigm type of the affix.
- Whether the words of this class can take both suffix and prefix or only one of them.
- The number of rules in this class.

Here, `SFX 1001 Y 17` states that this is a suffix class with paradigm type "1001" which has 17 rules, and these rules can be used even if the entry has prefixes.

The fields in the affix rules give the information on the characters to strip from the word, the affix to add to the word and the condition. For example, the first affix rule in Table 1 says, in the words with this flag ("1001"), strip ◌िस् from the end and add the suffix ◌ीः to form a valid word. The `condition` field is a regex-like expression that is checked from the end of a word for a suffix rule and from the start for a prefix rule (Németh, 2018). If there is nothing to be stripped from the word, the stripping is written as "0" (i.e. `SFX <flag> 0 <suffix> <condition>`). And if no suffix is to be added to the word, it is represented with "0" (i.e. `SFX <flag> <stripping> 0 <condition>`).

In the dictionary file, the affix flag "1001" is assigned to the word "आशिस्" (Table 1). So, the first affix rule strips ◌िस् from the end of "आशिस्" to form "आश" (note that it does not have a *virama*). Then, the rule adds ◌ीः to the end of "आश" to form "आशीः" which is the nominative singular form of "आशिस्" and therefore, a valid word.

The affix file is not only for defining affix rules. It is also used for containing a lot of options that help improve Hunspell's spellchecking process for the language which would be discussed later (Sections 5.2 and 5.3). In the following section, we describe the preparation of the dictionary and affix files.

5 Design and Preparation of the Dictionary

For designing the dictionary, we follow the word and paradigm model, which is computationally easy to work with. We prepare the dictionary entries and affix rules for nouns and verbs using two different methods, as we would describe in the next section (Section 5.1). Sections 5.2 and 5.3 discuss the use of some of Hunspell's options for improving the suggestions and handling optional characters.

5.1 Preparation of Words and Affix Rules

5.1.1 Nouns

Sanskrit has only two morphological classes at the inflectional level viz. noun and verb. In almost all the nouns, only the last few letters of the nominal base change when it combines with a nominal suffix. For example, the base, नदी, when it combines with the nominative dual suffix औ its final vowel ई is replaced by य् and becomes नद्यौ. Similarly, when भगवत् combines with instrumental dual suffix भ्याम्, the final त् is replaced by द् and becomes भगवद्भ्याम्. For such forms, the affix rules are written with the letters needed to be removed from the *base*, in the stripping field, and the suffix to be added, in the suffix field (Table 2). But

Form	Stripping	Suffix	Affix Rule
नदी	◌ी	◌ौ	SFX 1 ◌ी यौ .
भगवद्भ्याम्	त्	द्भ्याम्	SFX 2 त् द्भ्याम् .
रामस्य	0	स्य	SFX 3 0 स्य .
हरिः	0	◌ः	SFX 4 0 ◌ः .
सीतायाः	0	याः	SFX 5 0 याः .

Table 2: Affix rules for different *subanta* forms

in some cases, the base remains unchanged in its declined form, such as रामस्य, हरिः, गुरुभ्याम्, सीतायाः, भगवत्सु, etc. For such forms, the affix rules are written with "0" in the stripping field (Table 2).

The affix rules cannot be framed for some of the noun forms that completely differ from their bases (irregular forms). For example, दुह् becomes धुक् in nominative singular[9] and अस्मद् becomes मम in genitive singular. These being exceptional cases, the corresponding forms are directly added to the dictionary. Similarly, the indeclinables, which do not have any suffixes, are also directly added to the dictionary without any affix flags.

5.1.2 Verbs

The verb morphology is much more complex than the noun morphology. The verbs in certain tenses or moods have both prefix as well as suffix. Further, in a certain tense (*liṭ lakāra*) there is a reduplication of the verbal stem. We do not discuss the morphological details at length here due to the constraints in the page size. However, the general morphological structure of the finite verb forms is:

UPASARGA (prefix) + **A/Ā**[10] (prefix) + **DHĀTU** (root) + **VIKARAṆA** (tense marker) + **TIṄ** (verbal suffix)

We have here two prefixes and two suffixes. But Hunspell supports[11] only one prefix and two suffixes for a word (Németh, 2018). Hence, we merge A/Ā-prefix with the UPASARGA-prefix to form a single prefix. Also, we merge the root with the tense marker to form the stem thereby reducing the suffix to one.

Due to this limitation of the Hunspell to use at the most a single prefix and at the most a single suffix, we have to transfer the load of base formation under different environments to the dictionary, thereby resulting in more than one stem corresponding to each verb. For example, the root, स्पर्ध् has two entries in the dictionary corresponding to it – स्पर्ध and पस्पर्ध. With regard to the affix rules for the finite verb forms, they are almost same for a given tense or mood.

5.2 Improving Suggestions

For Hunspell to accurately suggest correct alternatives for an incorrect word, it needs a list of characters used in the script and a list of common misspellings. We provide these lists using the `TRY` and `REP` options, respectively, in the affix file.

5.2.1 Try Characters

Using the `TRY` characters, Hunspell can suggest the correct words when the misspelled words differ from them by a single character (Németh, 2018). Hunspell adds, deletes, or replaces one of these characters to suggest the closest valid dictionary word for the misspelled word (Shepelev, 2021).

`TRY` characters are more effective if they are in the order of their frequency in the literature (Németh, 2018; Shepelev, 2021). For this, we use the Amarakośa of Amarasiṃha which contains the frequently used words in classical Sanskrit and calculate the frequency of the characters[12]. We, then, add the characters to the `TRY` option in the descending order of their frequency as shown below:

```
TRY ◌् ◌ा र त ◌ि क स न व य म प ◌ु ◌ः द ◌ो
◌े ल ◌ं ◌ी श ष ग च ण ध ह ज भ ◌ौ ◌ृ ◌ू थ ऽ
ब ट ङ ड ◌ै अ ख ञ घ ठ छ आ उ फ इ ऋ ढ ए झ
ऊ ओ ई ऐ औ[13]
```

[9]दादेर्धातोर्घः *Aṣṭādhyāyi 8/2/32* and एकाचो बशो भष् झषन्तस्य स्ध्वोः *Aṣṭādhyāyi 8/2/37.*

[10]Only in some tenses/moods (*laṅ, luṅ* and *lṛṅ lakāras*).

[11]Hunspell also supports two prefixes and one suffix when the COMPLEXPREFIXES option is set. But it is mainly used by languages with a right-to-left writing system.

[12]The algorithm and the results are available at: https://github.com/vipranarayan14/sanskrit-char-frequency

[13]Space-separated for readability

5.2.2 Replacement Definitions

Replacement definitions are provided for handling typical spelling mistakes (Németh, 2018). Based on these replacement definitions, Hunspell makes some replacements in the misspelled word to find the valid forms from the dictionary and suggests the same. Let us see an example. Consider the word "रामह". It is not a morphologically valid word. We know that the closest valid alternative is "रामः". But, using only the `TRY` characters, Hunspell will suggest many correct alternatives of which "रामः" is not even among the first three (Figure 1). This is where the `REP` (replacement) definitions become very helpful.

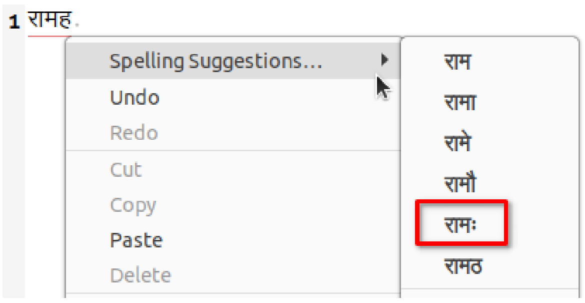


Figure 1: Correct suggestion in the 5th position.

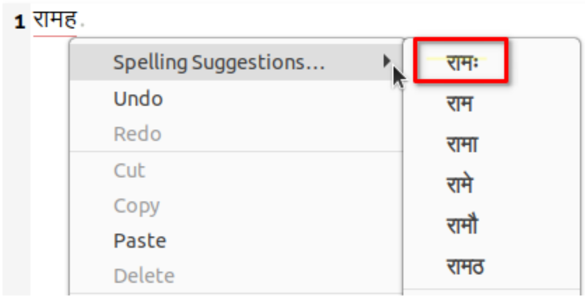


Figure 2: Correct suggestion in the 1st position.

When the replacement definitions shown in the Table 3 are added to the affix file we can see in Figure 2 that "रामः" becomes the first suggestion for the misspelled word "रामह". This is also because Hunspell gives `REP` suggestions the highest priority in the suggestion list (Németh, 2018; Shepelev, 2021).

First line in the replacement table (Table 3) is the **header**. It tells that there are 125 `REP` (replacement) definitions in the table. The rest of the table consists of the replacement definitions. We use the replacement definitions for suggesting closer valid alternatives

REP 125	
REP ह ◌ः	REP व ब
REP म् ◌ं	REP ब व
REP ओं ओ	REP य थ
REP ◌ि ◌ी	REP थ य
REP अ आ	REP ध घ
REP आ अ	REP घ ध
REP क ख	...
REP श ष	
...	

Table 3: Some of the replacement definitions used in the dictionary.

for words that are misspelled due to phonetic similarity or visual similarity.

The reasons for adding replacement definitions for **phonetically similar** characters are:

- Beginners in Sanskrit would not be familiar with the spellings, especially, of words that have some of the phonetically similar characters, such as श-ष in शेष, क-ख in कर, न-ण in बाणेन. Sanskrit has a lot of such words.

- People who are not trained with the INSCRIPT keyboard layout would tend to use the phonetic keyboard layouts for writing Sanskrit. So, their typing errors, for the most part, would also be phonetic.

- Phonetic typing errors also occur if the writer is not familiar with the input schemes such as ITRANS, Velthuis, Harvard-Kyoto, etc. since these popular schemes are themselves based on the Sanskrit phonemes. In Figure 2, the user, intending to write "रामः" using an ITRANS phonetic keyboard layout, has typed "`raamaha`" which resulted in "रामह" (since the key for *visarga* in ITRANS scheme is "H"; not "ha"). Now, the spellchecker using the replacement definitions, correctly suggests the right alternative "रामः".

The reasons for adding replacement definitions for **visually similar** characters are:

- They can help in correcting spelling errors found in post-OCR documents. For example, an OCR incorrectly recognises the word “मिथ्या” as “मिय्या” due to the visual similarity between “थ” and “य”. The spellchecker using the replacement definitions can help the user to correct it to “मिथ्या”.

- Users using Devanagari keyboards such as the INSCRIPT keyboard make spelling errors more due to the visual similarity of characters than their phonetic similarity.

5.3 Handling Optional Characters

There are many other options in Hunspell for improving the overall working of the spellchecker. One such option is `IGNORE` which can be used to make Hunspell ignore certain characters when spellchecking. Using this option, we ignore the character “ऽ” *(avagraha)* for allowing words such as “काऽपि” and “इतोऽपि” to be written without it as “कापि” and “इतोपि”. Because both forms are considered correct and both are in vogue.

In this section, we described the preparation of the spellchecking dictionary. The following section briefly describes the development of a web interface for Sanskrit spellchecking.

6 Web Interface

A web interface[14] is developed using the spellchecking dictionary and Hunspell, mainly, for supporting platforms such as Android and iPhone where Hunspell integration is currently not available. However, it can be used in desktop browsers also. Sanskrit text in Devanagari script can be typed in or pasted into the editor. The editor marks the incorrect words with a red underline. If an underlined word is clicked, a pop-up window opens next to the word showing a list of correct suggestions (Figure 3). When a suggestion is clicked, the incorrect word is replaced with the suggested word and the pop-up closes.

7 Evaluation

After the Hunspell dictionary was prepared, all the forms of the word paradigms were tested against it with the help of Nodehun[15], a Node.js binding for the Hunspell library, and Mocha[16], an automated JavaScript testing framework. In the test, all the forms were recognised as correct spellings.

To evaluate the spellchecking dictionary, a test corpus was prepared from three random OCR-ed pages of Śrīmad Vālmīki Rāmāyaṇa from the Sanskrit Wikipedia[17]. The *sandhis* and *samāsas* in the corpus were manually split since the dictionary does not support such words. The corpus consisted of 751 unique words.

The dictionary was added to Hunspell and, using a script written in Python, each word in the corpus was tested with the Hunspell spellchecking interface. The Hunspell dictionaries prepared by Kumar (2017) and Líbera (2018) were also tested in a similar manner for comparison. The results were then manually analysed.

Dictionary	Words	AC	AI	RC	RI
A	751	720	31	501	250
B	751	720	31	27	724
C	751	720	31	5	746

Table 4: Comparison of results of dictionaries A, B and C. AC = Actual correct words, AI = Actual incorrect words, RC = Words recognised by the dictionary as correct, and RI = Words recognised by the dictionary as incorrect.

Table 4, shows the comparison of results of the three dictionaries, viz. dictionary (A) proposed in this paper, (B) the one prepared by Kumar and (C) which was developed by Líbera. (A) performed very well compared to (B) and (C). The main reason for the poor performance of (B) and (C) was their limited vocabulary. (B) has only 3228 words in the dictionary file while (C) has only 838 Devanagari words. Though (B) uses affix rules, not many forms are supported by them. These affix rules were generated by the `affixcompress` tool[18] provided by Hunspell and are not grammar-based[19].

In the case of (A), all the words declared as correct were also actually correct implying a

[14]It can be accessed at: https://sanskrit-spellchecker.netlify.app

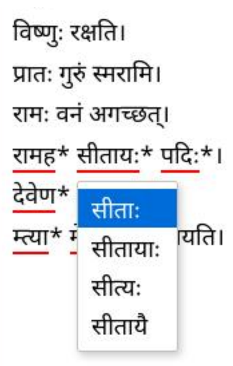
[15]https://www.npmjs.com/package/nodehun

[16]https://mochajs.org/

[17]https://sa.wikisource.org/s/2idh

[18]The tool is used for creating a spellchecking dictionary for Hunspell out of a list of words – https://github.com/hunspell/hunspell/.

[19]https://github.com/Shreeshrii/hindi-hunspell/issues/1#issuecomment-282244359



विष्णुः रक्षति।

प्रातः गुरुं स्मरामि।

रामः वनं अगच्छत्।

रामह* सीतायः* पदिः*।

देवेण*

म्त्या* म यति।

Figure 3: A screenshot of the web interface. All the words marked with * are incorrect. The editor correctly marks them with a red underline. For the incorrect word "सीतायः", the correct form "सीतायाः" is suggested.

100% precision. A few real-word errors were observed but they were not considered for this evaluation as Hunspell cannot handle such errors. Of the words reported as incorrect, a large percentage were wrongly considered incorrect (false negatives). The reasons were mainly traced to incomplete vocabulary and incorrect split of *sandhi* and *samāsa*. Since the dictionary is still in development false negatives due to out-of-vocabulary words are expected. Moreover, for better evaluation and for increasing the accuracy of suggestions, we are in the process of creating a larger test corpus.

The web interface was also manually tested using random words and sentences. The editor correctly marked the misspellings and also suggested correct and proper alternatives for them (Figure 3).

8 Observations

Some of the observations noted while preparing and evaluating the spellchecking dictionary are discussed here.

- **Use of Devanagari:** Though Unicode Devanagari is neither fully alphabetic nor syllabic in nature, but a combination of the two, and Pāṇini's word formation and euphonic rules operate at the alphabetic level, we decided to write the rules for Devanagari rather than other more suitable transliteration schemes such as WX or SLP1, that provide a one-to-one mapping between the Sanskrit phonemes and its romanised representation. The reason for this is, by doing so, we can avoid an intermediary step of converting the input into roman transliteration and then, converting the suggestions back into Devanagari which would require close interaction with the Hunspell APIs. But this would also mean, developing an altogether fresh set of rules if somebody uses IAST for Sanskrit.

- **Word-and-paradigm model:** The paradigm-based approach is advantageous for the preparation of the Hunspell dictionary for Sanskrit as it reduces the number of dictionary entries. Even though Sanskrit is a highly inflectional language, Pāṇini's grammar helps to reduce the forms of most of its words to a few hundred paradigms. We need to make affix rules only for these few hundred paradigms. For all other words having similar forms we just have to add the affix flag associated with the paradigm. This reduces the dictionary's size and at the same time, makes it easy to maintain. This approach also simplifies the process of adding new words. For example, if a new word, "कामदेव" is to be added, we add it to the dictionary file along with the flag associated with its paradigm, "देव". This way, we are able to add support for not only कामदेव but also all its declined forms without much effort.

- **Handling compounds:** The spellchecker's performance drops when there are compounds in the text. Though there are options for writing compound rules in Hunspell, it would not be possible to support all the compounds since Sanskrit is highly productive in terms of compound generation. Therefore, it is better to pipeline the user input into an existing *sandhi-samāsa* splitter like that of Hellwig's (2018) or Huet's (2005) before spellchecking it. This would again require close integration with the Hunspell APIs. Further, there are some words which cannot be handled even if the text is preprocessed with a *sandhi-samāsa* splitter. For example, the ending of some words changes in their compounded forms – राजा becomes राज in महाराज and आत्मा becomes आत्म in आत्मदुश्चरितः. These forms are valid only within the compound and not otherwise. To overcome this problem, such compounded words have to be directly added to the dictionary.

- **Spelling variations:** Words with *anusvāra* (ं) such as संभ्रमः and असंबाधम् optionally undergo homo-organic nasalisation and become सम्भ्रमः and असम्बाधम्, respectively. Both the forms are frequently used alternatively in the literature. So both forms had to be added to the dictionary. Also, the letter म् at the end of word is replaced by *anusvāra* if it is followed by a consonant[20]. So, for suffixes ending with म्, extra affix rules had to be created with *anusvāra* in the place of म्.

- **Context-dependent spelling errors:** Some of the spelling variations are context-dependent. For example, as mentioned above, the *anusvāra* at the end the word is valid only if the next word starts with a consonant. Similarly, verb forms which have the *a/ā*-prefix in the conditional mood, will drop it when they are preceded by the word मा[21]. For example, मा भैषीः and मा भवान् कार्षीत् instead of मा अभैषीः and मा भवान् अकार्षीत्, respectively. Such cases cannot be handled by the Hunspell since it only looks at one word at a time. Further, as is true with any spellchecker, it is impossible to detect wrong words in a given context that are correct otherwise (Deorowicz and Ciura, 2005). For example, use of ताम् instead of वाम्, where both the words are correct spelling-wise.

9 Conclusions and Future Work

The paper proposed and described the design and preparation of a Hunspell dictionary for Sanskrit. It also discussed the advantages of the paradigm-based approach which was followed for the generation of the dictionary. The proposed dictionary is grammar-based unlike that of Kumar (2017) and is general-purpose, that is, it can be used for correcting both typing errors and OCR errors. The paper also presented an online spellchecking interface for Sanskrit.

The current limitations of the dictionary and future improvements to be made are discussed below.

- The spellchecking dictionary is a work in progress. Currently, it supports nouns, indeclinables, pronouns and numbers (cardinals and ordinals). It also supports active and middle verb forms. Complex verb forms (*sannantas*), non-finite verb forms (*kṛdantas*), participles, etc. and secondary nominal derivatives (*taddhitas*) are yet to be supported.

- Hunspell's `KEY` option improves suggestions for words misspelled due to the proximity of letters on the keyboard. This can be utilized for INSCRIPT and phonetic keyboard layouts.

- Sanskrit is also written in many roman transliteration schemes such as IAST, ITRANS, etc. The current spellchecking dictionary works only for Devanagari. Special spellchecking dictionaries have to be made at least for the popular transliteration schemes of Sanskrit.

- Extensions/add-ons for more platforms/softwares/operating systems have to be developed so that users can enjoy the benefits of the dictionary in their favourite writing tools and environments.

Acknowledgements

I thank Dr. S. Muthu for his valuable support and encouragement. I thank Dr. Amba Kulkarni for all the resources and guidance. If not for Victor Shepelev's detailed blogs on Hunspell library (Shepelev, 2021), I could not have understood its format and inner workings. His detailed descriptions and clear-cut examples for the Hunspell format made it easy for me to develop the dictionary. Finally, I would like to thank the reviewers for their valuable feedback and suggestions.

[20]मोऽनुस्वारः *Aṣṭādhyāyi 8/3/23*

[21]न माङ्योगे *Aṣṭādhyāyī 6/4/74* and स्मोत्तरे लङ् च *Aṣṭādhyāyī 3/3/176*

References

Devaraja Adiga, Rishabh Kumar, Amrith Krishna, Preethi Jyothi, Ganesh Ramakrishnan, and Pawan Goyal. 2021. Automatic Speech Recognition in Sanskrit: A New Speech Corpus and Modelling Insights. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 5039–5050, Online. Association for Computational Linguistics.

Devaraja Adiga, Rohit Saluja, Vaibhav Agrawal, Ganesh Ramakrishnan, Parag Chaudhuri, K. Ramasubramanian, and Malhar Kulkarni. 2018. Improving the learnability of classifiers for Sanskrit OCR corrections. In *Computational Sanskrit & Digital Humanities: Selected Papers Presented at the 17th World Sanskrit Conference*, Vancouver, Canada.

Akshar Bharati, Vineet Chaitanya, and Rajeev Sangal. 2002. Natural Language Processing: A Paninian Perspective.

Sebastian Deorowicz and Marcin Ciura. 2005. Correcting spelling errors by modelling their causes. *Int. J. Appl. Math. Comput. Sci*, 15:275–285.

Veena Dixit, Satish Dethe, and Rushikesh K Joshi. 2016. Design and Implementation of a Morphology-based Spellchecker for Marathi, an Indian Language. *International Journal of Science Technology and Engineering*, 3(2):92–96.

Marcis Gasuns. 2013. Sanskrit-Hunspell. `http://samskrtam.ru/sanskrit-hunspell/`.

Oliver Hellwig and Sebastian Nehrdich. 2018. Sanskrit Word Segmentation Using Character-level Recurrent and Convolutional Neural Networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2754–2763, Brussels, Belgium. Association for Computational Linguistics.

Gérard Huet. 2005. A functional toolkit for morphological and phonological processing, application to a Sanskrit tagger. *Journal of Functional Programming*, 15(4):573–614. Publisher: Cambridge University Press.

Amita Jain, Minni Jain, Goonjan Jain, and Devendra K. Tayal. 2018. "UTTAM" An Efficient Spelling Correction System for Hindi Language Based on Supervised Learning. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 18(1):8:1–8:26.

Shailza Kanwar, Manoj Sachan, and Gurpreet Singh. 2017. N-Grams Solution for Error Detection and Correction in Hindi Language. *International Journal of Advanced Research in Computer Science*, 8(7):667–670.

Baljeet Kaur and Harsharndeep Singh. 2015. Design and Implementation of HINSPELL -Hindi Spell Checker using Hybrid approach. *International Journal of Scientific Research and Management*, 3(2). Number: 2.

Amba Kulkarni. 2009. *Anusaaraka: An approach for MT taking insights from the Indian Grammatical Tradition.* PhD, University of Hyderabad, Hyderabad.

Amba Kulkarni. 2019. *Sanskrit Parsing: Based on the Theories of Shabdabodha.* D.K. Print World Ltd, Shimla : New Delhi.

Amba Kulkarni and Devanand Shukl. 2009. Sanskrit Morphological analyzer: Some Issues. *Bh. K Festschrift volume by LSI.*

Shreedevi Kumar. 2017. Sanskrit Hunspell. `https://github.com/Shreeshrii/hindi-hunspell/tree/master/Sanskrit`. Original-date: 2014-12-03.

Aadil Lawaye and Bipul Syam Purkayastha. 2016. Design and Implementation of Spell Checker for Kashmiri. *International Journal of Scientific Research*, 5:199–200.

Gurpreet Singh Lehal. 2007. Design and Implementation of Punjabi Spell Checker. *International Journal of Systemics, Cybernetics and Informatics.*

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.

S Rajashekara Murthy, Vadiraj Madi, Sachin D, and Ramakanth Kumar P. 2012. A Non-Word Kannada Spell Checker Using Morphological Analyzer and Dictionary Lookup Method. *International Journal of Engineering Sciences & Emerging Technologies*, 2(2):43–52.

László Németh. 2018. Hunspell – Format of Hunspell dictionaries and affix files. `https://github.com/hunspell/hunspell/releases/download/v1.7.0/hunspell5.pdf`.

László Németh. 2019. Hunspell: About. `http://hunspell.github.io/`.

Dhaval Patel. 2016. SanskritSpellCheck. `https://github.com/drdhaval2785/SanskritSpellCheck`. Original-date: 2014-10-04.

Dhaval Patel. 2021. Hunspell for Sanskrit? - Issue #91 - sanskrit-lexicon/COLOGNE. `https://github.com/sanskrit-lexicon/COLOGNE/issues/91`.

Shabana Pathan, Nikhil Khuje, and Punam Kolhe. 2019. A Survey on Creation of Hindi-Spell Checker to Improve the Processing of OCR. *International Journal of Research in Engineering, Science and Management*, 2(3):517–519.

Amrutanshu Pradhan and Sasanka Sekhar Dalai. 2020. Design of Odia Spell Checker with word Prediction. *International Journal of Engineering Research & Technology*, 8(1). Publisher: IJERT-International Journal of Engineering Research & Technology.

Adriana Quintanilha and Vinícius Della Líbera. 2018. Sanskrit Spell Checker. `https://addons.mozilla.org/en-US/firefox/addon/sanskrit-spell-checker/`.

Carsten Schnober, Steffen Eger, Erik-Lân Do Dinh, and Iryna Gurevych. 2016. Still not there? Comparing Traditional Sequence-to-Sequence Models to Encoder-Decoder Neural Networks on Monotone String Translation Tasks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1703–1714, Osaka, Japan. The COLING 2016 Organizing Committee.

Jananie Segar and Sarveswaran Kengatharaiyer. 2015. Contextual spell checking for Tamil Language. In *14th Tamil Internet Conference*, Singapore.

Victor Shepelev. 2021. Rebuilding the spellchecker: Hunspell and the order of edits. `https://zverok.space/blog/2021-01-28-spellchecker-5.html`.

Namrata Tapaswi, Dr Suresh Jain, and Mrs Vaishali Chourey. 2012. Morphological-based Spellchecker for Sanskrit Sentences. *International Journal of Scientific & Technology Research*, 1(3):1–4.

Uma Maheshwar Rao G., Amba P. Kulkarni, Christopher Mala, and Parameshwari K. 2012. Telugu Spell-checker. *Center for Applied Linguistics and Translation Studies University of Hyderabad Hyderabad, India.*

TeQuAD: Telugu Question Answering Dataset

Rakesh Kumar Vemula, Manikanta Sai Nuthi and **Manish Shrivastava**
Language Technologies Research Center
International Institute of Information Technology
Hyderabad, India
{rakesh.kumar,mani.kanta}@research.iiit.ac.in and
m.shrivastava@iiit.ac.in

Abstract

Recent state of the art models and new datasets have advanced many Natural Language Processing areas, especially, Machine Reading Comprehension tasks have improved with the help of datasets like SQuAD (Stanford Question Answering Dataset). But, large high quality datasets are still not a reality for low resource languages like Telugu to record progress in MRC. In this paper, we present a Telugu Question Answering Dataset - TeQuAD with the size of 82k parallel triples created by translating triples from the SQuAD. We also introduce a few methods to create similar Question Answering datasets for the low resource languages. Then, we present the performance of our models which outperform baseline models on Monolingual and Cross Lingual Machine Reading Comprehension (CLMRC) setups, the best of them resulting in an F1 score of 83 % and Exact Match (EM) score of 61 %.

1 Introduction

MRC is one of the key tasks in NLP, where we test the ability of machines to understand and answer the questions using provided textual knowledge. In common Machine Reading Comprehension tasks, for a given query, the machine needs to extract the answer from the context (paragraph) in the form of span indices. A popular large-scale annotated reading comprehension dataset - SQuAD Rajpurkar et al. (2016), revolutionised the research interest in this area for English. And though decent research work has been done in MRC for a few Indian languages, for languages like Telugu, which is a Dravidian language, still need similar resources for such Natural Language Understanding task.

Creating an RC dataset of good quantity & quality is difficult, requires manpower, and is time-consuming. For a few languages, the dataset is created by translating SQuAD and using few matching techniques to extract the span indices of answers in the target language (Carrino et al. (2019), Abadani et al. (2021), Artetxe et al. (2019)). For others, the dataset is created by using the methodology followed in the creation of SQuAD (Lim et al. (2019), Efimov et al. (2020), Cui et al. (2018), d'Hoffschmidt et al. (2020)).

Our idea is to introduce a few heuristics based approaches to create the datasets for a low resource language (Telugu) using the resources from a high resource language via translation. An obvious challenge is to extract the span of the answers in the translated Contexts. With translation, due to language divergences, the position and structure of the answer in the context will vary in the translated language, making it difficult to use straight-forward approaches, like translation candidate matching, to find the position of the answer in the context. We focused on the span extraction process, which is crucial for such a dataset creation after translation. We applied these methods to SQuAD v1.1 and created TeQuAD, a MRC dataset for Telugu consisting of 82k parallel Telugu-English triples (Paragraphs, Questions, and Span indices of Answers). The intention to create a parallel dataset is to exploit the advantage of Cross-lingual reasoning. We also introduce a supervised approach to extract the span of the most probable answer from the target paragraph. This span extractor can later aid in data augmentation for MRC in low-resource languages. In cases where the heuristics do not work, our supervised method performs better than the matching techniques due to its ability to consider contextual semantic information using pre-trained language models.

Both monolingual and cross-lingual setups of multilingual Bidirectional Encoder Representations from Transformers (BERT) were trained on TeQuAD and evaluated on TiDyQA (Clark et al., 2020) and on two other test datasets, which we created manually by correcting a few samples from translated SQuAD and by using Wikipedia articles respectively.

Our dataset and code are available here[1].

2 Related Work

Several datasets such as SQuAD(Rajpurkar et al., 2016), NewsQA dataset (Trischler et al., 2016) and CNN/Dailymail (Chen et al., 2016), etc fulfills the necessity of resources in English for QA tasks. Although these datasets helped in attaining enormous progress for this specific language in NLP, other languages are still unexplored in this area due to the scarcity of high-quality annotated datasets in corresponding languages. While the generation of reading comprehension corpora in other languages is costly and time-consuming, few works such as Lim et al. (2019), Efimov et al. (2020), Cui et al. (2018), d'Hoffschmidt et al. (2020) developed RC datasets natively. Clark et al. (2020) presented a question answering dataset covering 11 typologically diverse languages including Telugu.

Few others propose methods to boost the functioning of the model in low-resource settings. Hsu et al. (2019) explored zero-shot cross-lingual transfer learning on reading comprehension tasks and suggested that translation from source to target languages is not necessary.

Bornea et al. (2020) presents translation-based data augmentation mechanism to improve multilingual transfer learning.

Liu et al. (2020) and Cui et al. (2019) talks about leveraging translated information from the high resource languages to perform well in low resource languages. Cui et al. (2019) presented several back translational approaches for cross-lingual experiments. They have also discussed techniques to align the answer phrases in the target language. Stating the disadvantages of such approaches and the necessity to overcome them, they introduced a novel model called 'Dual BERT', which has the ability to learn semantic information from bilingual QA pairs and utilize the learned knowledge to improve MRC in low resource languages.

Yuan et al. (2020) introduced phrase boundary supervision tasks to improve the answer boundary detection capability in the low resource MRC models which are trained with training data from high resource languages to exploit cross-lingual transfer learning.

Post correction methods to improve the span of the extracted answer are addressed in Reddy et al. (2020). They added additional layers on top of a pre-trained transformer-based language model to re-examine and modify the predicted answers.

3 Corpus Creation

A simple and cost-efficient technique to create a dataset for an NLP task is to translate a well-annotated existing dataset. When it comes to MRC tasks, SQuAD is favorite for its quality and adaptability to recent implementations of deep learning models. This span extractive QA data is created from English wikipedia articles by crowd workers. More than 100000 triples were generated in SQuAD1.1. A triple consists of a Question, an Answer to the question in the form of span indices, and a Context where the answer can be found.

We translated the English SQuAD triples to the Telugu language using online Google translator[2], obtaining translated triples consisting of translated Telugu paragraphs, questions, and answers.

After translation, a well-known issue is difficulty in extraction of the span indices of the translated answers. Considering the different possibilities of translated Telugu answer phrase's presence in the translated Telugu context, we followed multiple techniques to extract the span for answer phrases. The purpose of following different techniques is to create as much synthetic data as possible for Telugu MRC. We also present a supervised span extraction technique to handle the cases where rule-based methods fail.

3.1 Matching

We used matching algorithms like cosine similarity and fuzzy search with a threshold value of greater than 0.7. A window sliding through the translated Telugu context computes the matching score between the phrase inside the window and the translated Telugu answer phrase. Samples are considered if such a matching phrase (matching score greater than the threshold) is found in the translated Telugu context, else ignored. There might be a possibility of the presence of multiple answer phrases in the context. For such samples, we considered the index of the actual English answer phrase among its repetitions present in English context and selected the corresponding index as the answer from repetitions of the Telugu answer in translated Telugu context. For example, if the word 'apple' is present 3 times in the English context, and if the answer is the second repeated instance, then we consider the

[1]https://github.com/rakeshvemula1157/TeQuAD

[2]https://translate.google.co.in/

	English	**Telugu (ISO 15919)**
Context	China Mobile had more than "**2,300**" base stations suspended due to power disruption or severe telecommunication "**traffic congestion**". Half of the wireless communications were lost in the Sichuan province . China Unicom 's service in Wenchuan and four nearby counties was cut off , with more than 700 towers suspended.	Vidyuttu antarāya lēdā tīvramaina ṭelikamyūnikēṣan "**ṭrāphik raddī**" kāraṇaṅgā cainā mobail "**2,300**" ki paigā bēs'sṭēṣanlanu nilipivēsindi. Sicuvān prānslō saga vairles kamyūnikēṣanlu pōyāyi. Vencuvān mariyu samīpanlōni nālugu kauṇṭīlalō cainā yunikām sēva nilipivēyabaḍindi, 700ki paigā ṭavarlu nilipivēyabaḍḍāyi.
Question 1	Besides power disruption , what caused telecommunications to be suspended ?	Vidyuttu antarāyantō pāṭu, ṭelikamyūnikēṣanlanu nilipivēyaḍāniki kāraṇamēmiṭi?
Span	16 - 17	5 - 6
Answer	traffic congestion	ṭrāphik raddī
Question 2	How many base stations are suspended?	Enni bēs sṭēṣanlu saspeṇḍ cēyabaḍḍāyi?
Span	5 - 5	10 - 10
Answer	2,300	2,300

Table 1: Representation of QA pairs in parallel corpora

second repetition of the Telugu word (Āpil) as the answer in the translated Telugu context.

3.2 Explicit Position Indicator

We managed to extract the answer span for few ignored samples by marking the English answer phrase before translating to Telugu. English answer phrase in the English context is marked by the special symbol ('|') and then translated to the Telugu language. The marked symbol ('|') remains unchanged, making it easier to find the translated Telugu answer phrase in the translated Telugu context.

Using these approaches, we were able to obtain 82,605 English-Telugu parallel triples - creating a reading comprehension dataset, TeQuAD. Table 1 shows the representation of parallel corpora of English - Telugu. For evaluation, we have created two different test datasets to analyze the performance of models on Telugu MRC.

- **Translated & Corrected dataset:**
  1000 English triples from the dev set of SQuAD1.1 are translated to Telugu and corrected manually. A set of guidelines is prepared and explained in 3.4 to correct the translated Telugu context, questions, and answers.
- **Wiki dataset:**
  Similar to SQuAD, we created this data from Wikipedia articles. Randomly selected wikipedia articles are splitted into paragraphs. From 125 Telugu Wikipedia paragraphs, 947 QA pairs are created manually by framing questions with answer types such as Person, Location, Date/Time, Quantities, Clauses, Verb phrases, Adjective phrases and others. Minimum 5 and maximum 10 questions were created for each paragraph/context.

3.3 Span Extractor

In TeQuAD, we obtained the span indices for the Telugu answers by using the above-mentioned matching techniques. Such rule-based techniques might not provide better results in cases where,

1. The translated Answer might not be present in the translated Context (information about the answer might have lost or different form of the answer generated in translation). See Figure 1 for example.
2. Multiple instances of the translated Answer might be in the translated Context. See Figure 2 for example.
3. A partial answer phrase returns a better matching score with the translated answer phrase than the actual answer phrase. See Figure 3 for example.

Example 1
English Context:
. . . Trade liberalization may shift economic inequality from a global to a domestic scale . . .
English Question:
What scale does trade liberalization shift economic inequality from ?
English Answer:
Global
Translated Telugu Context:
. . . వాణిజ్య సరళీకరణ ఆర్థిక అసమానతలను ప్రపంచ స్థాయి నుంచి దేశీయ స్థాయికి మార్చవచ్చు . . .
(. . . Vāṇijya saraḷīkaraṇa ārthika asamānatalanu prapan̄ca sthāyi nun̄ci dēśīya sthāyiki mārcavaccu . . .)
Translated Telugu Question:
వాణిజ్య సరళీకరణ ఏ స్థాయి నుండి ఆర్థిక అసమానతను మారుస్తుంది ? (Vāṇijya saraḷīkaraṇa ē sthāyi nuṇḍi ārthika asamānatanu mārustundi)
Plausible Telugu Answer: ప్రపంచ స్థాయి (prapan̄ca sthāyi)
Translated Telugu Answer: గ్లోబల్ (Glōbal)

Figure 1: Example for absence of translated Answer in the translated Context. Both *’prapanca sthāyi’* and *’glōbal’* share the similar meaning.

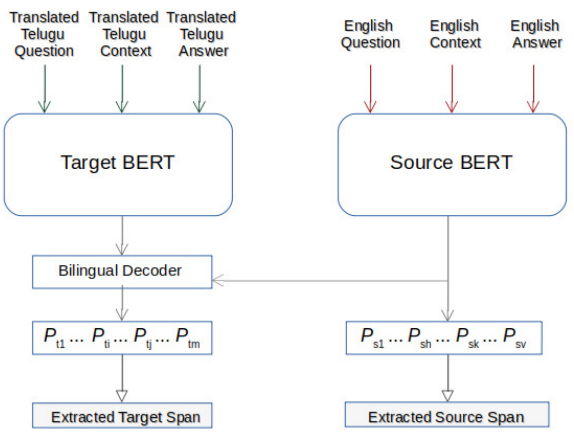
Translated Telugu Context:
. . . పూర్తిగా పెట్టుబడిదారీ ఉత్పత్తి పద్ధతిలో కార్మికుల వేతనాలు ఈ సంస్థలు లేదా యజమాని ద్వారా నియంత్రించబడవు, కానీ మార్కెట్ ద్వారా, వేతనాలు ఏ ఇతర మంచి కోసం ధరల మాదిరిగానే పనిచేస్తాయి. అందువలన, వేతనాలు నైపుణ్యం యొక్క మార్కెట్ ధర యొక్క విధిగా పరిగణించబడతాయి . . .
(. . . Pūrtigā peṭṭubaḍidārī utpatti pad'dhatilō kārmikula vētanālu ī sansthalu lēdā yajamāni dvārā niyantrin̄cabaḍavu, kānī mārkeṭ dvārā, vētanālu ē itara man̄ci kōsaṁ dharala mādirigānē panicēstāyi. Anduvalana, vētanālu naipuṇyaṁ yokka mārkeṭ dhara yokka vidhigā parigaṇin̄cabaḍatāyi . . .)
Translated Telugu Question:
పూర్తిగా పెట్టుబడిదారీ ఉత్పత్తి పద్ధతిలో వేతనాలను ఏది నియంత్రిస్తుంది ? (Pūrtigā peṭṭubaḍidārī utpatti vidhānanlō vētanālanu ēdi niyantristundi ?)
Translated Telugu Answer:
మార్కెట్ (mārkeṭ)

Figure 2: Example for multiple instances of Answer in the Context

Translated Telugu Context:
. . . కొంతమంది చట్టపరమైన అవిధేయతలు సామాజిక ఒప్పందం యొక్క చెల్లుబాటుపై వారి విశ్వాసం కారణంగా శిక్షను అంగీకరించడం తమ బాధ్యత అని భావిస్తారు . . .
(. . . Kontamandi caṭṭaparamaina avidhēyatalu sāmājika oppandaṁ yokka cellubāṭupai vāri viśvāsaṁ kāraṇaṅgā śikṣanu aṅgīkarin̄caḍaṁ tama bādhyata ani bhāvistāru . . .)
Translated Telugu Question:
చట్టపరమైన అవిధేయతలు దేనిపై విశ్వాసం కారణంగా శిక్షను అంగీకరిస్తారు ?
(Caṭṭaparamaina avidhēyatalu dēnipai viśvāsaṁ kāraṇaṅgā śikṣanu aṅgīkaristāru ?)
Plausible Telugu Answer: సామాజిక ఒప్పందం యొక్క చెల్లుబాటుపై (Sāmājika oppandaṁ yokka cellubāṭupai)
Translated Telugu Answer: సామాజిక ఒప్పందం యొక్క ప్రామాణికత (Sāmājika oppandaṁ yokka prāmāṇikata)

Figure 3: Example for partial matching answer scenario.

In order to handle such cases, we introduce a supervised method to extract span indices for the translated answers. We use the Dual BERT approach proposed in Cui et al. (2019), but along with parallel QA pairs, we also pass their parallel answers as input to the model and the span indices of the answers are predicted (See Figure 4). Due to its ability to exploit semantic information from both Telugu-English parallel triples, it can identify a modified variant of answer phrase in the translated context, even if the translated answer phrase does not present in the translated context completely.



Translated Telugu Question
Translated Telugu Context
Translated Telugu Answer
English Question
English Context
English Answer
Target BERT
Source BERT
Bilingual Decoder
$P_{t1} \cdots P_{ti} \cdots P_{tj} \cdots P_{tm}$
$P_{s1} \cdots P_{sh} \cdots P_{sk} \cdots P_{sv}$
Extracted Target Span
Extracted Source Span

Figure 4: Architecture of Span Extractor

Unlike the above-mentioned matching techniques, this model can identify the correct instance of the answer in the translated context, even if there are multiple instances present. In addition to the translated Telugu answers, information from English Answers will help the model to retrieve span indices of the complete Telugu answer phrases in the translated contexts.

As this is a supervised method that needed training data, we considered 82k parallel triples from TeQuAD consisting of span indices obtained by using matching techniques, for training. For evaluation, we use the Translated & Corrected test dataset where span indices of the Telugu answers are manually corrected. We pass the translated Telugu answers as input to the model and evaluate the predictions with corrected Telugu answers. The experimental setup is similar to the Cross-lingual section. Results attained show the performance of **88%** F1 Score and **73%** EM Score. Besides apparent advantages, such supervised methods needs sufficient resources to perform well and predicts a span even if the answer information is not present in the context (e.g. might have been lost in machine translation).

3.4 Manual QA Correction

We have used Google NMT for translating triples. Although Google NMT is efficient and the quality of the translations is good, the present translation machines are not smart enough to generate accurate translations for low resource languages like Telugu. After a translation is generated by the machine, there has to be a human correction to ensure the piece of translation is grammatically correct, comprehensible, and carries the exact information present in the English text (by deducting/appending the knowledge obtained/lost from the translation of the English text.)

3.4.1 Correction Guidelines

Data in SQuAD is in triples form. All three components of triples (para, question, and answer) are translated from English to Telugu. The order followed to correct a translated triple is:

- Correct the Telugu paragraph.
- Correct the Telugu question according to the Telugu paragraph.
- Extract the Telugu answer according to the Telugu question from the Telugu paragraph.

While correcting a paragraph or a question, two essential aspects should be considered:

- **Adequacy:** The meaning/knowledge provided in the English para/question should be preserved in the Telugu para/questions.
- **Fluency:** The structure/syntax of the Telugu para/question should be proper/readable.

Answers obtained by translation are partial, in a few cases incorrect. Such answers should be corrected based on the corresponding Corrected Telugu Questions, Corrected Telugu Paras, and English Answers. The answer should be obtained from its paragraph (Corrected Telugu Para) and must be recorded. By using these guidelines, we corrected 1000 samples which are used as test set.

4 MRC Experiments

We experimented on TeQuAD in monolingual and cross-lingual setups. The pre-trained Multilingual-BERT (mBERT) trained in 104 languages including Telugu and English is employed for obtaining encoded representations for both languages. We use nltk tokenizer followed by BERT Word Piece tokenizer to sub tokenize the tokens in all the experiments. Experimented with a batch size of 64 and sequence length of 512. As in Google's Tensorflow implementation of BERT, ADAM with weight decay optimizer is considered with different learning rates for different experimental setups. Our models have been trained on Google Cloud TPU v2.

Monolingual setup: In the monolingual setup, 82k Telugu triples from TeQuAD are considered for fine-tuning the mBERT model for MRC task. We used Google's Tensorflow implementation of BERT for running SQuAD tasks and trained it for 3 epochs with the learning rate of 1e-4.

Cross-lingual setup: The dual BERT approach proposed in Cui et al. (2018) is used for the CLMRC setup. In this approach, deep contextualized representations of the inputs from both languages are considered and 'Bilingual Context' is computed, which will be used to exploit the semantic relations among the English and Telugu QA pairs. Parallel QA pairs of English and Telugu are passed as inputs to the model and span indices of the Telugu answer phrases are predicted. 82k Parallel Telugu-English triples from TeQuAD are considered for fine-tuning the pre-trained mBERT model. We used the implementation in Cui et al. (2019) and trained it for 3 epochs with the learning rate of 2e-5.

Both the Cross-Lingual and Monolingual fine-tuned models are evaluated on three test datasets. Along with Translated & Corrected (1000) and Wiki (947) test datasets, Telugu samples of Gold Passage task (Span Extractive QA task) from TyDiQA dev (667) dataset are considered for evaluation.

F1 score and EM score are used as evaluation metrics. Results of the evaluation for monolingual and cross-lingual setups are shown in Table 2.

5 Results and Observations

The most important observation from the results is that the models fine tuned on TeQuAD performed way better than the zero-shot mBERT model. On average, a 40% increase in F1 and EM scores were obtained in all setups.

Our experiments also show that performance on the Wiki test dataset is better than others. It must be noted that Wiki test dataset is well annotated, created from Telugu Wikipedia articles, and has better quality than the Translated & Corrected test dataset. In contrast, the quality of the TyDiQA Telugu samples is low and not recommended for fair Telugu MRC evaluation. Most of the queries in TyDiQA revolve around the area of lands, zip codes, date of births/deaths *etc.* Learning on a set of similar types of questions more often will make the model over-

Model	Test Dataset	Mono-Lingual		Cross-Lingual	
		F1	EM	F1	EM
mBERT(Zero shot)	Translated & Corrected	28.4	0.0	27.1	0.01
	Wiki QA	27.1	0.0	27.6	0.0
	TyDi dev QA	21.0	0.0	21.3	0.0
mBERT(TeQuAD)	Translated & Corrected	69.4	43.7	69.4	43.5
	Wiki QA	**83.0**	**61.0**	**83.3**	**61.9**
	TyDi dev QA	61.0	41.6	69.1	43.3

Table 2: Experimental results of MRC on Test Datasets. Performance (in terms of %) F1 : F1 Score and EM: Exact Match Score

Test Dataset	**TyDi QA**		**TeQuAD**	
	F1	EM	F1	EM
Translated-&-Corrected	57.7	29.5	69.4	43.7
Wiki QA	77.3	48.4	83.0	61.0

Table 3: Comparison b/w TeQuAD and TyDi QA for Telugu MRC. Performance (in terms of %) F1: F1 Score and EM: Exact Match Score

fit on these types of questions, but would lack the ability to comprehend other types. As expected, the model trained on TyDiQA train dataset achieved good performance when evaluated on TyDiQA dev dataset, but the performance fell behind compared to TeQuAD model when evaluated on Translated-&-Corrected and Wiki test datasets.

Why low EM scores ?

Although decent F1 scores have been registered, the gap between the two metrics is notable. The difference between the metrics is approximately 20% across all the setups. We tried to analyze the reasons for error predictions. One reason for the low Exact Match score is multiple possible answers for a query. Different answers, all that seems to be correct might affect the MRC model generating the exact answer. See Figure 5 for Example.

Another obvious reason for faulty answer predictions is the low-to-moderate resources available for the language. Pre-trained models exposed to such fewer data resources might not be able to reason the context leading to false answer predictions. And even though such models leverage the information from high resource language(s), due to the linguistic divergences between the languages (here Telugu and English), answer boundary detection capability in the low resource language is poorer, failing to identify the complete answer phrase in the context.

In Yuan et al. (2020), they discussed the deficient answer boundary detection capability of MRC models for low resource languages. Their work suggested improving the detection capability by training the MRC model on phrases in low-resource language, mined from the internet. We experimented by mining approximately 32k Telugu phrases from Wikipedia and trained the model with the phrase masking prediction task. Results don't show any noticeable improvement in the EM scores.

On the other hand, several MRC works employ character-level span indices to point the answer

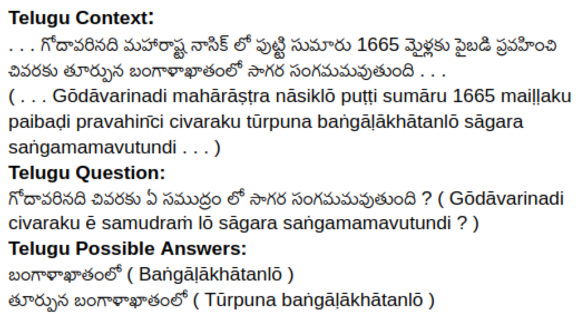

Telugu Context:
. . . గోదావరినది మహారాష్ట్ర నాసిక్ లో పుట్టి సుమారు 1665 మైళ్లకు పైబడి ప్రవహించి చివరకు తూర్పున బంగాళాఖాతంలో సాగర సంగమమవుతుంది . . .
(. . . Gōdāvarinadi mahārāṣṭra nāsiklō puṭṭi sumāru 1665 maiḷḷaku paibaḍi pravahiñci civaraku tūrpuna baṅgāḷākhātanlō sāgara saṅgamamavutundi . . .)
Telugu Question:
గోదావరినది చివరకు ఏ సముద్రం లో సాగర సంగమమవుతుంది ? (Gōdāvarinadi civaraku ē samudraṁ lō sāgara saṅgamamavutundi ?)
Telugu Possible Answers:
బంగాళాఖాతంలో (Baṅgāḷākhātanlō)
తూర్పున బంగాళాఖాతంలో (Tūrpuna baṅgāḷākhātanlō)

Figure 5: Example for multiple possible answers

Experimental Setup	**Translated & Corrected**		**Wiki QA**	
	F1	EM	F1	EM
Mono-lingual	65.9	39.1	79.3	50.5
Cross-lingual	67.4	39.7	82.2	54.0

Table 4: Results of the Experimental setups trained on less corpora : 34k QA pairs. Performance (in terms of %) F1: F1 Score and EM: Exact Match Score

phrase specifically. This might lead to worse EM scores in Telugu considering the rich morphology of the language. So, instead, we followed word-level span indices for the answer phrases.

Why Cross-lingual Experimentation?

As Discussed, Cui et al. (2019) proposed Dual BERT approach to improve the MRC for low resource languages by utilizing cross-lingual knowledge. With experiments, we observed that CLMRC setup helps in boosting the performance of the model when the size of the corpora is low (See 4). But with the creation of large synthetic data, the effect of CLMRC setup is negligible. In table 2, results obtained by training the model on 82k data in mono-lingual setup are identical to the results of CLMRC setup. Creation of such resources helps the machine to learn from the target language itself instead of relying on High resource languages.

Comparison with TyDiQA

Clark et al. (2020) presents the performance of Gold-Passage MRC (Similar to SQuAD style QA) in Telugu. They train the model on approximately 49k Multilingual QA pairs and evaluate it on the Telugu test dataset. We also experimented by fine-tuning mBERT on TyDIQA 49k QA pairs and evaluated it on the above-mentioned test datasets. See table 3 for a comparison between models trained on TyDiQA and TeQuAD. The TeQuAD based model outperformed the TyDiQA trained model in Telugu MRC.

6 Conclusion and Future work

As a move towards the creation of the QA dataset for Indian languages, this work took a step forward in the MRC corpus creation using translation. To record decent performances in NLP tasks for low resource languages, sufficient resources are necessary. As we discussed, the creation of such resources is difficult. Resources from high resource languages like English might be considered for the creation of datasets for low-resource languages like Telugu to create abundant data for different NLP tasks.

We introduce creation/correction techniques for such datasets improving the quality along with the quantity of the datasets along with providing mechanisms for further augmenting data. In the future, we would like to improve the MRC task for Telugu, provide a collection of pre-trained models trained on openly available resources in the Telugu language, as well as create additional data resources for the Telugu language.

References

Negin Abadani, Jamshid Mozafari, Afsaneh Fatemi, Mohammd Ali Nematbakhsh, and Arefeh Kazemi. 2021. Parsquad: Machine translated squad dataset for persian question answering. In *2021 7th International Conference on Web Research (ICWR)*, pages 163–168. IEEE.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2019. On the cross-lingual transferability of monolingual representations. *arXiv preprint arXiv:1910.11856*.

Mihaela Bornea, Lin Pan, Sara Rosenthal, Radu Florian, and Avirup Sil. 2020. Multilingual transfer learning for qa using translation as data augmentation. *arXiv preprint arXiv:2012.05958*.

Casimiro Pio Carrino, Marta R Costa-jussà, and José AR Fonollosa. 2019. Automatic spanish translation of the squad dataset for multilingual question answering. *arXiv preprint arXiv:1912.05200*.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858*.

Jonathan H Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. Tydi qa: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2019. Cross-lingual machine reading comprehension. *arXiv preprint arXiv:1909.00361*.

Yiming Cui, Ting Liu, Wanxiang Che, Li Xiao, Zhipeng Chen, Wentao Ma, Shijin Wang, and Guoping Hu. 2018. A span-extraction dataset for chinese machine reading comprehension. *arXiv preprint arXiv:1810.07366*.

Martin d'Hoffschmidt, Wacim Belblidia, Tom Brendlé, Quentin Heinrich, and Maxime Vidal. 2020. Fquad: French question answering dataset. *arXiv preprint arXiv:2002.06071*.

Pavel Efimov, Andrey Chertok, Leonid Boytsov, and Pavel Braslavski. 2020. Sberquad–russian reading comprehension dataset: Description and analysis. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 3–15. Springer.

Tsung-Yuan Hsu, Chi-Liang Liu, and Hung-yi Lee. 2019. Zero-shot reading comprehension by cross-lingual transfer learning with multi-lingual language representation model. *arXiv preprint arXiv:1909.09587*.

Seungyoung Lim, Myungji Kim, and Jooyoul Lee. 2019. Korquad1. 0: Korean qa dataset for machine reading comprehension. *arXiv preprint arXiv:1909.07005*.

Junhao Liu, Linjun Shou, Jian Pei, Ming Gong, Min Yang, and Daxin Jiang. 2020. Cross-lingual machine reading comprehension with language branch knowledge distillation. *arXiv preprint arXiv:2010.14271*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Revanth Gangi Reddy, Md Arafat Sultan, Efsun Sarioglu Kayi, Rong Zhang, Vittorio Castelli, and Avirup Sil. 2020. Answer span correction in machine reading comprehension. *arXiv preprint arXiv:2011.03435*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*.

Fei Yuan, Linjun Shou, Xuanyu Bai, Ming Gong, Yaobo Liang, Nan Duan, Yan Fu, and Daxin Jiang. 2020. Enhancing answer boundary detection for multilingual machine reading comprehension. *arXiv preprint arXiv:2004.14069*.

A Comprehensive Study of Mahabharat using Semantic and Sentiment Analysis

Ranjani Parthasarathi, Professor and
Aishwarya Kumaran and Nithyasri and Shanmuga Priya and Srijeyarankesh J S
College of Engineering Guindy
Anna University, Chennai- 600 025

Abstract

Indian epics have not been analyzed computationally to the extent that Greek epics have. In this paper, we show how interesting insights can be derived from the ancient epic Mahabharata by applying a variety of analytical techniques based on a combination of natural language processing methods like semantic analysis, sentiment analysis and Named Entity Recognition (NER). The key findings include the analysis of events and their importance in shaping the story, character's life and their actions leading to consequences and change of emotions across the eighteen parvas of the story.

1 Introduction

Semantic analysis is the study of the meaning of language, whereas sentiment analysis is the study of emotions that has been depicted. Sentimental analysis is prevailing in various domains such as social media monitoring, customer support management, and analysing customer feedback.

Mahabharat is a tightly interwoven story with intricate characters traversing various incidents resulting in many course of actions. This makes Mahabharat an interesting study for analysing such characters and incidents using the various NLP techniques. The Mahabharat Ganguli translation is used for conducting such an analysis. Entity Analysis involves named entity recognition which helped discover many unfamiliar characters present in Mahabharat. Semantic Analysis is used to analyze sentence structure to highlight the events and their resulting actions whereas sentiment analysis is used to analyze the flow of emotions as the story progresses. Character analysis describes the character's life, the trails and tribulations the character has been through and his/her characteristics. The paper presents a unified technique to achieve the above as stated.

2 Related Works

Mahabharatha is an epic with valuable lessons on life and values .Epics like Mahabharata are a kind of tragedy and are built around noble men within the form of narratives. A tragedy typically has a plot with a beginning, a middle and an end and other constituents of the text are secondary to the plot. The start of the plot typically is a scenario of stability which gets disturbed by some events. Plots of tragedy have various constituents i.e. suffering, reversal, recognition of latest knowledge, surprise. An epic is different from a newer literary genre like a novel and will have lot of negative sentiment across its breadth but in spite of that conveys a noble theme in the minds of its audience.

Debabrati et al. (Das et al., 2016) has proposed the usage of NLP techniques such as sentiment analysis and characterization of important characters with respect to their emotion. Mabhabharatha text is tokenized using standard NLP techniques. - The tokens are POS (parts of speech) tagged and tagged tokens are mapped to synsets in Wordnet in a word sense disambiguation process. - The sentiment scores are picked up from SentiWordnet for each synset. - Overall sentiment of the parva is derived from these values by summing the constituent sentiment scores. Emotion analysis for the full text and each of the protagonists is done with the help of NRC word-emotion association lexicon. After extracting the relevant part of the corpus,the score is calculated for each POS (part of speech) tagged token for each emotion and finally summed up. However, by this approach one cannot get an overall view of the character in terms of their life, relations and actions but only about their emotions. The usage of lexicon based approach limits the ability of the model to learn new vocabulary. The proposed idea in this paper aims to remove these two limitations.

Named Entity Recognition is identifying proper-nouns in the text. The biggest challenge in Named Entity Recognition is the lack of sufficient labelled data. This poses a challenge for NER in Mahabharat as the standard tagged datasets are different in comparison. Active Learning is an efficient option as it helps identify samples that will be the most informative to the model(Li et al., 2022), discuss active learning technique for Named Entity Recognition. Further work was done by Yanyao Shen et.al (Shen et al., 2017) where a CNN-CNN-LSTM model was built for NER, in an iterative approach. They used the various selection strategies for NER such as least confidence, Maximum Normalized Log-Probabilities.

Named Entity Recognition is a sequence labelling task.(Akhundov et al., 2018) discusses the merits of using Bidirectional Long Short Term (BiLSTM) models for sequence labelling tasks. For any sequence labelling task the model is required to take into consideration the context of the entire sentence.

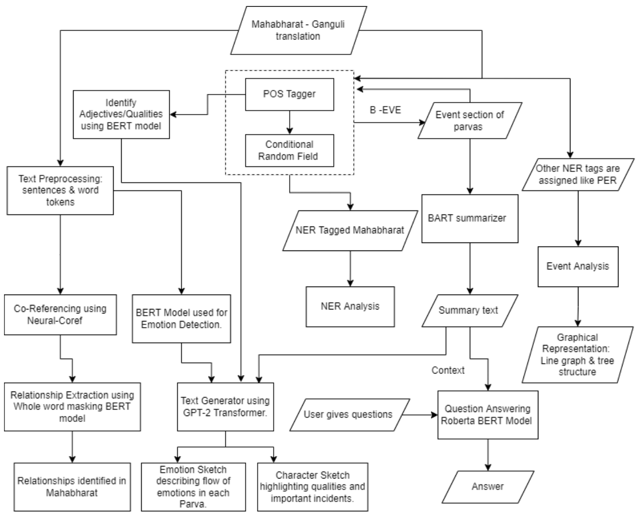
(Devlin et al., 2019) introduced a model called BERT. BERT was trained on two tasks - masked word prediction and next sentence prediction. These tasks can make use of data that requires no labelling and is widely available.

Conditional random field is a popular probabilistic method for structured prediction.(Sutton and McCallum, 2010) discussed the problem of classification by predicting a single discrete class variable y given a vector of features.

In co-referencing resolution, training recurrent neural networks to model long term dependencies is an issue faced.(Dhingra et al., 2017) had proposed to use external linguistic knowledge as an explicit signal to inform the model which memories it should utilize.

3 Methods

This section describes the design and implementation of the system being proposed with the help of overall system architecture represented in Figure 1. In this section the proposed methodology is discussed. Using Natural Language Processing techniques such as co-referencing, relationship extraction, analysis on events and many other functions are performed like automated question-answering, graphical representations and identifying relationships of different entities in the Mahabharat dataset. Relationship extraction is a key task done with the help of co-referencing. Event analysis with the help of BART for summarization and BERT for question answering. The character sketch is drawn from using adjective extraction model using BERT and POS tags. The POS tags along with generated summary of each parva in Mahabharatha is used to draw the character sketch. The emotion sketch is derived from using BERT model by using emotions from Go-Emotion dataset. The generated summary along with emotions extracted in every parva is passed through a text generation model for generating an emotion sketch.

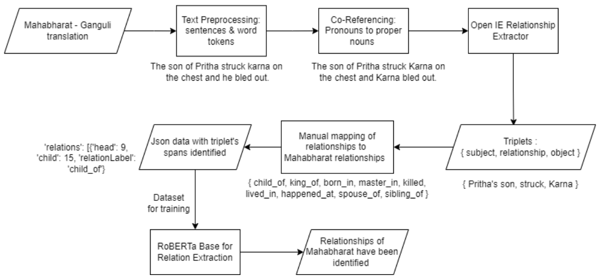


Mahabharat - Ganguli translation
POS Tagger
Identify Adjectives/Qualities using BERT model
Conditional Random Field
B -EVE
Event section of parvas
Other NER tags are assigned like PER
Text Preprocessing: sentences & word tokens
NER Tagged Mahabharat
BART summarizer
Event Analysis
NER Analysis
Co-Referencing using Neural-Coref
BERT Model used for Emotion Detection.
Summary text
Graphical Representation: Line graph & tree structure
Context
Relationship Extraction using Whole word masking BERT model
Text Generator using GPT-2 Transformer.
User gives questions
Question Answering Roberta BERT Model
Relationships identified in Mahabharat
Emotion Sketch describing flow of emotions in each Parva.
Character Sketch highlighting qualities and important incidents.
Answer

Figure 1: Overall Architecture Diagram

3.1 Raw Dataset

Kisari Mohan Ganguli's translation of the Sanskrit epic Mahabharat is the raw data acquired. The raw data consists of eighteen books. They are Adi Parva, Sabha Parva, Vana Parva, Virata Parva, Udyoga Parva, Bhishma Parva, Karna Parva, Shalya Parva, Sauptika Parva, Stri Parva, Santi Parva, Anusasana Parva, Aswamedha Parva, Asramavaisika Parva, Mausala Parva, Mahaprasthanika Parva and Svargarohanika Parva. The entire dataset has 1,35,850 sentences.



Mahabharat - Ganguli translation
Text Preprocessing: sentences & word tokens
Co-Referencing: Pronouns to proper nouns
Open IE Relationship Extractor
The son of Pritha struck karna on the chest and he bled out.
The son of Pritha struck Karna on the chest and Karna bled out.
'relations': [{'head': 9, 'child': 15, 'relationLabel': 'child_of'}
Json data with triplet's spans identified
Manual mapping of relationships to Mahabharat relationships
Triplets : { subject, relationship, object }
{ child_of, king_of, born_in, master_in, killed, lived_in, happened_at, spouse_of, sibling_of }
{ Pritha's son, struck, Karna }
Dataset for training
RoBERTa Base for Relation Extraction
Relationships of Mahabharat have been identified

Figure 2: Relationship Extraction Architecture Diagram

TAG	EXPANSION
B-PER	Beginning of Person entity
I-PER	Inside a Person entity
B-PLACE	Beginning of Place entity
I-PLACE	Inside a Place entity
B-EVE	Beginning of Event entity
I-EVE	Inside a Event entity
B-WEAPON & WAR STRATEGY	Beginning of Weapons and War Strategy entity
I-WEAPON & WAR STRATEGY	Inside a Weapons and War Strategy entity
B-COMMUNITY	Beginning of Communities entity
I-COMMUNITY	Inside a Communities entity
B-LIT and ART	Beginning of Literature and Art entity
I-LIT and ART	Inside a Literature and Art entity

Table 1: NER tags and their Expansions

Relation	Subject, Object
Child of	Person, Person
King of	Person, Place
Born in	Person, Place
Master of	Person Literature, Artifact
Killed	Person, Person/Place/Weapon
Lived in	Person, Place
Happened in	Event, Place
Spouse of	Person, Person
Sibling of	Person, Person
Friend of	Person, Person
Leader of	Person, Community/Place
Guardian of	Person, Person/Community
Belongs to	Person, Community/Place

Table 2: NER tags and their Expansions

3.2 Tool Used

The tools, libraries and environments used include (pandas development team, 2020), (Abadi et al., 2015), (Akbik et al., 2019), (Hunter, 2007), (Mausam et al., 2012), (Harris et al., 2020), (Pedregosa et al., 2011) and (Loper and Bird, 2002).

3.3 Named Entity Recognition

In this process, entities pertaining to Mahabharat have been identified as listed in the Table 1. To automatically identify these entities from the text, we trained a CRF model on the Mahabharat dataset. The Conditional Random Field model considers the semantics of the given text where, given a sequence of input words we obtain the sequence of output labels. Training set $\left\{\left(\mathbf{X}^{(t)}, \mathbf{y}^{(t)}\right)\right\}$ is a set of input and target sequences pairs:

input words are $\mathbf{X}^{(t)} = \left[\mathbf{x}_1^{(t)}, \ldots, \mathbf{x}_{K_t}^{(t)}\right]$
target labels are $\mathbf{y}^{(t)} = \left[y_1^{(t)}, \ldots, y_{K_t}^{(t)}\right]$

K_t is the length of the t^{th} sequence.
A set of features from the Mahabharat dataset has been crafted which is provided to the CRF model. The features of the sentence given to the model include the case of the word, the last few letters of the word. The implementation of the Conditional Random Field model has been motivated from the Sklearn-CRFSuite (Pedregosa et al., 2011). It has been modified based on the features for Mahabharatha text.

3.4 Relationship Extraction

The Relationship Extraction architecture is represented in Figure 2. which involves co-referencing and RoBERTa for relationship extraction.The process of co-referencing involves replacing the pronouns by their respective proper nouns in the sentence. For each mention or a pair of mentions a set of features are crafted. The most likely antecedent is mapped to its corresponding mention.After the co-referencing phase, the text has the proper noun in place of the pronoun referencing it. Coreference Resolution has been implemented through Neural Coref model. This co-referenced data is sent to the OpenIE model which finds all the relationships in the data. The output is given as a triplet of entities and the relationship identified. The relationship triplets identified here has to be filtered according to the relationships mentioned in Mahabharat. The dataset is analysed to identify fourteen relationships as listed in Table 2 between the entities identified in the Mahabharat text. The dataset with entities, relationship labels and its tokens are given to the RoBERTa base Model. The relationship extractor is thus trained on the given dataset.

3.5 Event Analysis

The event analysis architecture is represented in Figure 3. The important tasks involved in event analysis are summary generation, question - answering and graphical representation of the insights obtained.

Summary Generation :

After the Mahabharat dataset has been tagged

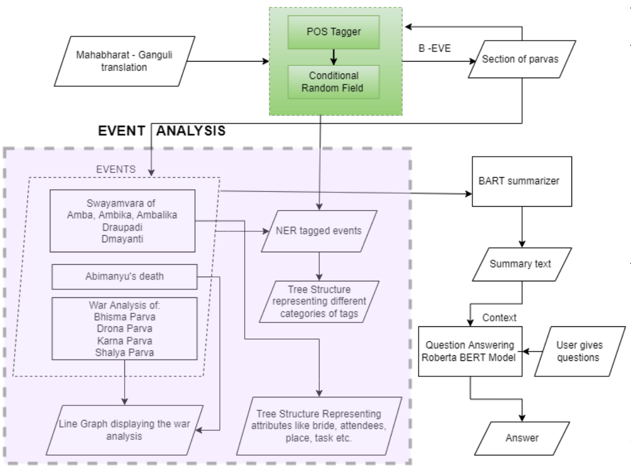

Mahabharat - Ganguli translation
POS Tagger
Conditional Random Field
B -EVE
Section of parvas
EVENT ANALYSIS
EVENTS
Swayamvara of Amba, Ambika, Ambalika Draupadi Dmayanti
Abimanyu's death
War Analysis of: Bhisma Parva Drona Parva Karna Parva Shalya Parva
NER tagged events
Tree Structure representing different categories of tags
BART summarizer
Summary text
Context
Question Answering Roberta BERT Model
User gives questions
Answer
Line Graph displaying the war analysis
Tree Structure Representing attributes like bride, attendees, place, task etc.

Figure 3: Event Analysis Architecture Diagram

by the NER tagger the section of the document describing the events are identified by the B-EVE and I-EVE tags. This sections of Parvas are given to the summary model to extract the summary of each event identified. The events identified include swayamvaras of Amba, Ambalika, Ambika, Draupadi and Damayanti, Abhimanyu's death and war analysis on different parvas. Different kinds of analysis are performed on the events and represented in graphs. The BART model generates summary. The embedding in a BART model is built on top of BERT. For every text sequence in its input, the BERT encoder outputs an embedding vector for each token in the sequence as well as an additional vector containing sentence-level information. The pre-training is done using the masked sequences. BART uses additional masking mechanisms as shown in Figure 4.

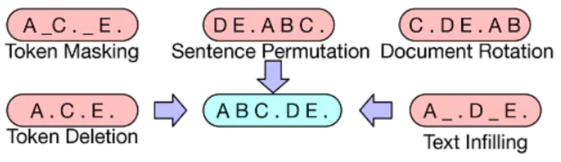


Figure 4: BART sentence masking

Question-Answering : The question-answering task is performed by whole word masking BERT model where the model gives the answer for the given question from the context. BERT model predicts the probability of each word being the starting and ending index of the answer span.
The BART model discussed in the previous phase outputs the summary of the event. The summary of the event is given to the question-answering model which identifies the answer span of data

TAG	EXPANSION
`Attendees`	People present at the event
`Chosen one`	The groom
`Bride`	The one who chooses
`Father of Bride`	King who organized the event
`Place it was held`	The kingdom
`Weapon used`	Weapons used in the event

Table 3: Template for the Swayamvara graph

from the context for the specific question given by the user. The fine tuning of the question and answering model was done using the SQuAD(Stanford Question Answering Dataset).

Visualization : The insights of events of Mahabharat obtained on characters involved, place of the event etc. is represented by a tree structure as shown in Figure 12 which helps in comparing the event. Table 3 shows the entities of the template.

3.6 Character Analysis

Character Analysis is done so as to present a holistic view of the character in perspective of Mahabharatha. It includes the qualities of the character, their relationships, trials and tribulations they have been through and consequences of their actions. The Figure 5 depicts the flow of execution in performing this task. The Qualities are extracted using the 11 POS tags i.e [ADJ],[PUNCT], [ADV], [INTJ], [NOUN], [PROPN], [VERB], [CCONJ], [NUM],[PART],[AUX]. The extracted relations and the generated summary are used to create the character sketch with the help of a text generation model.

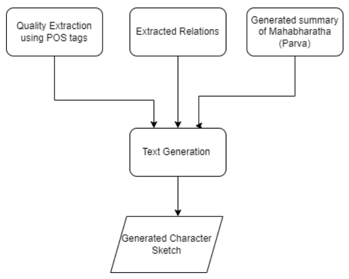

Quality Extraction using POS tags
Extracted Relations
Generated summary of Mahahbharatha (Parva)
Text Generation
Generated Character Sketch

Figure 5: Character Sketch Diagram

Quality Extraction : The Qualities of a person define who he/she is in the story. These are exhibited using adjectives in the story. The adjectives have to be extracted using POS tags using BERT model. The BERT model is already fine-tuned on the UPenn-Treebank dataset with an accuracy of about 97.25%. The top 15 adjectives are chosen by frequency corresponding to the character as they distinctly represent the character's qualities.

Summary Generation : The summary is generated using the BART model built on top of the BERT model. The input is given parva wise to the summary generation model so that necessary information is captures which can be later used for any generation tasks.

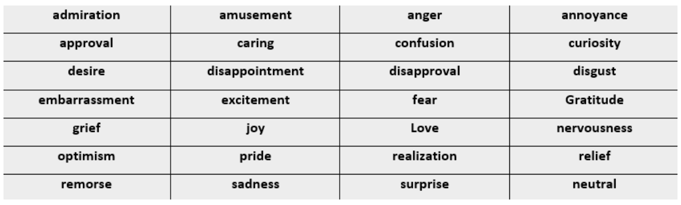
Text Generation : OpenAI's GPT-2 model is used for text generation. The GPT-2 transformer takes in a sequence of input tokens and then tries to generate multiple sequences of tokens in some chronological order so they form a meaningful sequence. The sequence of tokens generated are appended together to form a text. The Mahabharath summary alongwith the set of adjectives are taken as input collectively with some keywords such as "marriage", "parents", "born" etc. The model tries to decipher information related to these keywords and incorporates into the final text. Thus a character sketch is generated.

3.7 Emotion Analysis

Mahabharatha being an epic, contains a myraid of emotions throughout. It is important to identify these emotions and present them to the user in the most concise way possible without losing out information being captured. This is done by employing a emotion detection mechanism initially using a BERT model. This paper uses 26 different emotions as per the Go-Emotions dataset by Demsky et al. (Demszky et al., 2020). The extracted emotions are then fed to the text genration model collectively with the summarized text of Mahabharath (Parva-wise).

Emotion Detection : The Go-emotions dataset employs these 26 emotions as the necessary ones that can accurately capture different emotions while also not losing out on the context. BERT model is initially trained on an annotated parva of Mahabharatha with these set of emotions. The model is then deployed in the other 17 Parvas. Every sentence is attributed with some dominant emotion and the emotion which is dominant in one section of the parva is chosen as the right emotion. Every Parva contains about 100 sections and this procedure is followed for every Parva.

Text Generation : OpenAI's GPT-2 model is used for this text generation phase. The GPT-2 model takes in output from emotion detection phase and generated summary of Mahabharatha parva-wise. The keywords such as "feelings", "tension", "dilemma" are given as inputs alongwith the model so that the generated text is able to capture related incidents pertaining to those keywords. The text is presented to the user in the form of parapraghs.



admiration	amusement	anger	annoyance
approval	caring	confusion	curiosity
desire	disappointment	disapproval	disgust
embarrassment	excitement	fear	Gratitude
grief	joy	Love	nervousness
optimism	pride	realization	relief
remorse	sadness	surprise	neutral

Figure 6: Emotions used in the paper

TAG	COUNT
PERSON	1689
PLACE	173
EVENT	20
WEAPON and WAR STRATEGY	22
COMMUNITY	524
LIT and ART	23

Table 4: NER tags and their Expansions

4 Results

This section explains the results of semantic analysis tasks on Mahabharat like NER, Relationship extraction, Summary and Question Answering modules, Character Analysis and Emotion Analysis.

4.1 Entity Analysis

The entire text of Mahabharat has been annotated using a Conditional Random field model tuned for Named Entity Recognition.

Following this, a Relations Extraction model was built and the dataset for this model consists of parvas 5,6 of the Mahabharat text. The size of the training data is 2164 sentences. The model is capable of recognizing the relationships between the entities as one of the 14 categories as shown in Table 1. From the annotated data, the following inferences were made.

The number of unique entities in each category identified in the text are shown in Table 4. The frequency distribution of unique occurrences of each entity type is calculated and visualized as a pie chart as shown in Figure 7.
The "person" entities are then paired with each other based on their occurrences in the text. Two

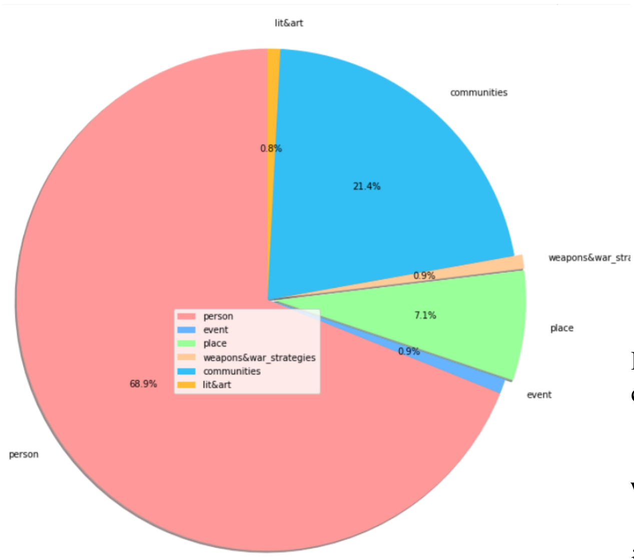
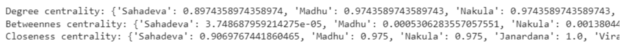


Figure 7: Distribution of Entities in the Text

person entities are said to be connected if they occur together in a span of 30 words. A network graph is thus constructed. The weight of the edges are assigned based the frequency of the particular pair. From this graph three different centralities are identified as shown in Figure 8.



```
Degree centrality: {'Sahadeva': 0.8974358974358974, 'Madhu': 0.9743589743589743, 'Nakula': 0.9743589743589743,
Betweennes centrality: {'Sahadeva': 3.748687959214275e-05, 'Madhu': 0.0005306283557057551, 'Nakula': 0.00138044
Closeness centrality: {'Sahadeva': 0.9069767441860465, 'Madhu': 0.975, 'Nakula': 0.975, 'Janardana': 1.0, 'Vira
```

Figure 8: Centralities

Each centrality represents a different kind of information about the entity.
Degree centrality is a measure of the number of other people a person is connected to. Higher the degree centrality, the more the person is connected.
Betweenness centrality is a measure of the popularity of the person. It is a measure of how many nodes are connected to others through this node.
Closeness centrality is a measure of the weightage of each of the connections in the graph. As the name indicates it shows how close each entity is to its neighbour.
The top fifteen entities with the highest closeness centralities are identified and a graph is plotted in the same manner as before, to show their interrelationships. Figure 9 shows that graph.

The entities included in this graph are those that have closest relationships with others. Arjuna has some of the most highly weighted edges implying that he is one of the most strongly connected character in the book. In addition this graph also shows the strong relationship between the five pandavas,

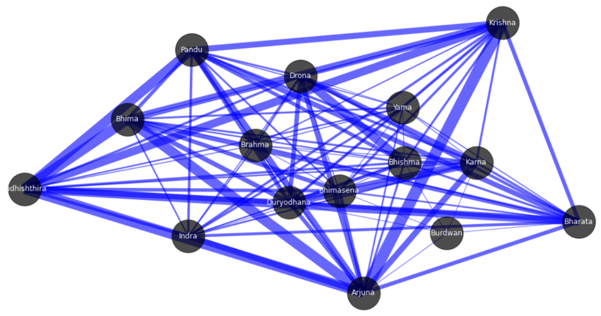


Figure 9: Interrelationships graph based on closeness centrality

with eachother and with Krishna.

In addition, the top fifteen entities based on their individual frequencies were identified and their interrelationships are represented in a network graph as shown in Figure 10.

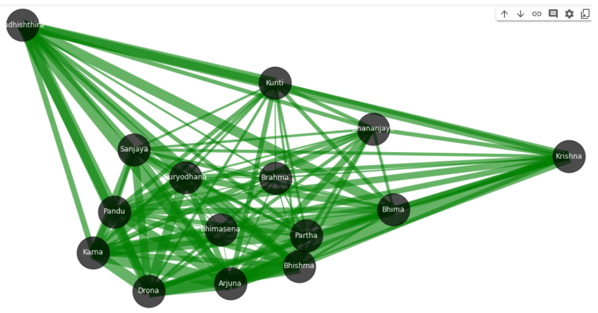


Figure 10: Interrelationships graph based on Frequency

This graph shows the most frequently occurring characters in the text. On comparison with the graph based on characters with highest centrality, two additional characters are identified - Sanjaya and Kunti. This shows that these two characters occur frequently in contexts outside of interactions.

4.2 Event Analysis

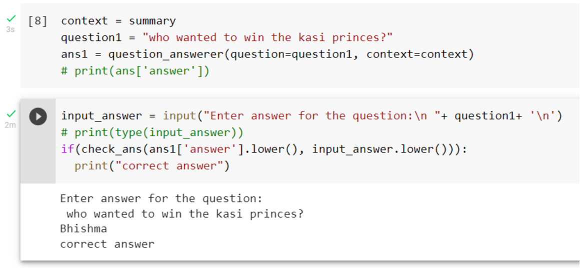
The entities tagged as events are identified and areas of the text where they are clustered are inferred to be the major events. These include Swayamvara, War parvas, Abimanyu's Death, dice game and disrobing. These events are analysed using Summary generation model, Question Answering models and through graphs.

4.2.1 Swayamvara

The Swayamvara event was analysed using the following tasks.

Summary Generation: The text pertaining to the event are fed as input to the summary generation model as a sequence of paragraphs. This model gives a 3-4 line output for the given input sample. The summary model is able to retain all important entity information and conveys the overall sequence of events in a succinct way.

Question Answering model for Quiz App: The output of the summary model is used as the context for the Question Answering model. The Q and A model has been used to build a quiz application, where the user is presented with an event and a set of questions pertaining to the question. The model identifies the answer from the context summary, and compares the answer it to the one given by the user. This has been demonstrated in Figure 11.



```
[8] context = summary
    question1 = "who wanted to win the kasi princes?"
    ans1 = question_answerer(question=question1, context=context)
    # print(ans['answer'])
```

```
input_answer = input("Enter answer for the question:\n "+ question1+ '\n')
# print(type(input_answer))
if(check_ans(ans1['answer'].lower(), input_answer.lower())):
  print("correct answer")
```

```
Enter answer for the question:
 who wanted to win the kasi princes?
Bhishma
correct answer
```

Figure 11: Snippet of the Quiz Application

In addition, there are provisions for the users to give their own questions to the model about each event.

Analysis and Graphical Representation: The event swayamvara is analysed using semantic graphs and a quiz app. A semantic graph with a fixed set of fields is defined for the events. By using the relationships identified in the event context, the values for the fields are filled.The template for the semantic graph of the event Swayamvara consists of the entity types mentioned in Table 3.This graph allows a comparison between the events. Figure 12 depicts the semantic graph for the Swayamvara of Amba, Ambika and Ambalika. This graph displays that the ceremony was held for three people together.

Figure 13 depicts the semantic graph for the Swayamvara of Panchali. The large number of attendess shows that a lot of important people took part in the competition.

4.3 War Analysis

The war events are distributed across four parvas such as Bhishma Parva, Drona Parva, Karna Parva, Shalya Parva. The performance of the Pandavas in each of these Parvas is plotted in a graph. If a member of the Pandava army is pierced or struck

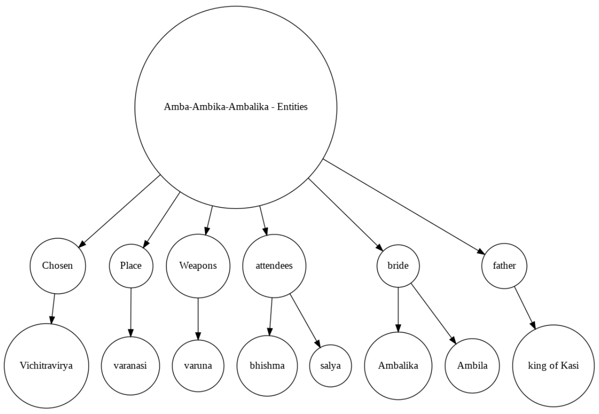


Figure 12: Semantic Graph depicting Amba's Swayamvara

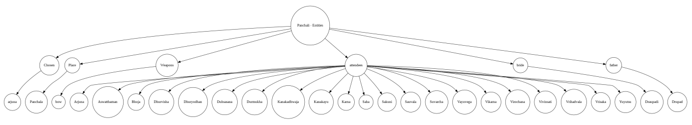


Figure 13: Semantic Graph depicting Panchali's Swayamvara

there is a small dip in the graph, if they are slayed a slightly bigger dip is shown and an even bigger dip is shown when they are slaughtered in bigger numbers. Similarly peaks of sizes proportional to the defeat of the Kouravas can be seen. Figure 14 and 15 show the graphs for the war parvas.

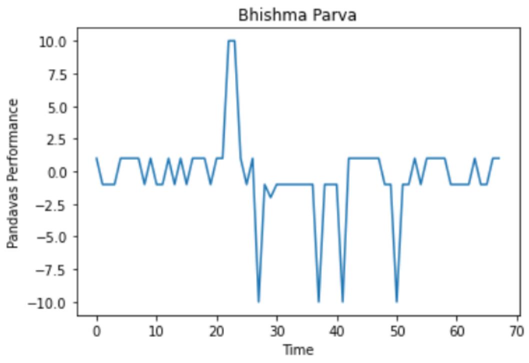


Figure 14: Line graph showing the performance of the Pandavas in Bhishma Parva

These graphs also allow us to track the battle sequence. The Kouravas saw major victories in Bhishma and Drona Parva, which is demonstrated by the major dips in the corresponding graphs. The victory of the Pandavas is shown in the final peak in the final graph.

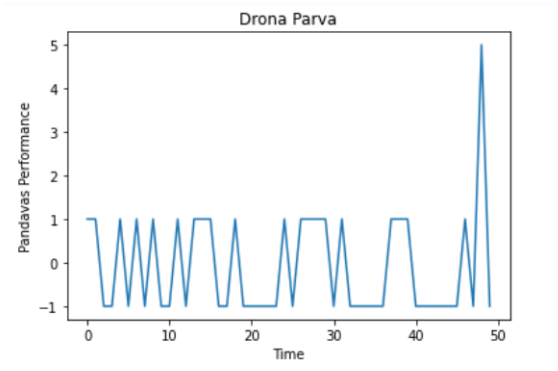


Figure 15: Line graph showing the performance of the Pandavas in Drona Parva

4.4 Abhimanyu's Death

The events surrounding Abhimanyu's death mark a turbulent battle between him and the kouravas. Abimanyu's efforts and performance at the time of his death are plotted in Figure 16. The graph shown in Figure 16 demonstrates how well Abhimanyu fought before the time of his death.

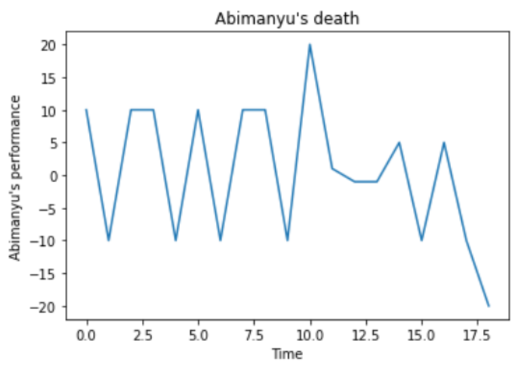


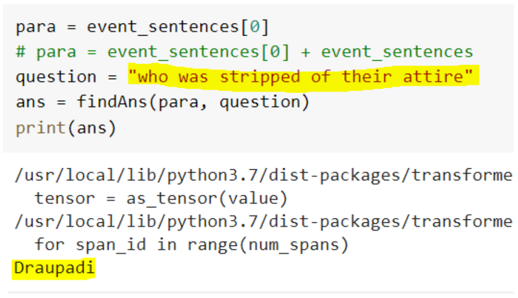
Figure 16: Line graph showing the performance of the Abhimanyu at the time of his death

4.5 Dice game and Disrobing

The dice game and disrobing event is analysed through summary generation and the q and a model and a quiz app is built for the same. Further there is also a provision for the users to ask questions related to a particular event as shown in Figure 17.

4.6 Character Analysis

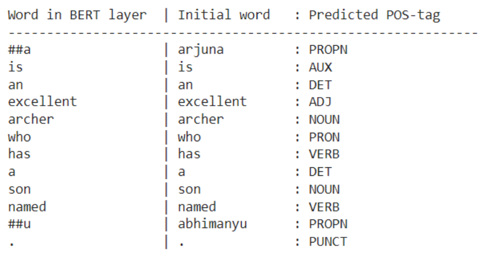
The Figure 18 depicts an example of how BERT identifies POS tags and extracts [ADJ] tags for adjectives. These adjectives are used alongwith summary to generate the character sketch. The user can utilise the character sketch to learn about the particular character's life, qualities and consequences



```
para = event_sentences[0]
# para = event_sentences[0] + event_sentences
question = "who was stripped of their attire"
ans = findAns(para, question)
print(ans)
```

```
/usr/local/lib/python3.7/dist-packages/transforme
  tensor = as_tensor(value)
/usr/local/lib/python3.7/dist-packages/transforme
  for span_id in range(num_spans)
Draupadi
```

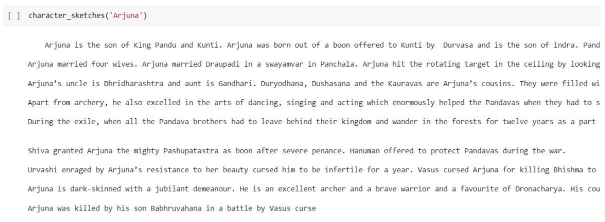
Figure 17: Q and A for disrobing event



```
Word in BERT layer  | Initial word   : Predicted POS-tag
-------------------------------------------------------------
##a                 | arjuna         : PROPN
is                  | is             : AUX
an                  | an             : DET
excellent           | excellent      : ADJ
archer              | archer         : NOUN
who                 | who            : PRON
has                 | has            : VERB
a                   | a              : DET
son                 | son            : NOUN
named               | named          : VERB
##u                 | abhimanyu      : PROPN
.                   | .              : PUNCT
```

Figure 18: An Example of a sentence with POS tags

of his/her actions instead of reading the entire text. The Figure 19 shows the final output of character sketch.



```
[ ]  character_sketches('Arjuna')
```

```
    Arjuna is the son of King Pandu and Kunti. Arjuna was born out of a boon offered to Kunti by  Durvasa and is the son of Indra. Pand
  Arjuna married four wives. Arjuna married Draupadi in a swayamvar in Panchala. Arjuna hit the rotating target in the ceiling by looking
  Arjuna’s uncle is Dhridharashtra and aunt is Gandhari. Duryodhana, Dushasana and the Kauravas are Arjuna’s cousins. They were filled wi
  Apart from archery, he also excelled in the arts of dancing, singing and acting which enormously helped the Pandavas when they had to s
  During the exile, when all the Pandava brothers had to leave behind their kingdom and wander in the forests for twelve years as a part

  Shiva granted Arjuna the mighty Pashupatastra as boon after severe penance. Hanuman offered to protect Pandavas during the war.
  Urvashi enraged by Arjuna’s resistance to her beauty cursed him to be infertile for a year. Vasus cursed Arjuna for killing Bhishma to
  Arjuna is dark-skinned with a jubilant demeanour. He is an excellent archer and a brave warrior and a favourite of Dronacharya. His cou
  Arjuna was killed by his son Babhruvahana in a battle by Vasus curse
```

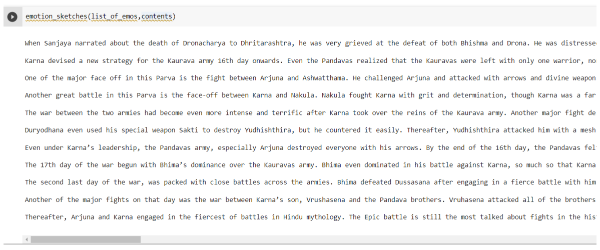
Figure 19: Character sketch of Arjuna

4.7 Emotion Analysis

The Emotion analysis analyzes the emotion sentence-wise and attributes the most occurring emotion to the section containing those sentences. The user can enter the Parva of choice for which the emotions are to be deduced. The emotions can be used as a basis for deriving any other analysis of the Mabhabaharatha text. The Figure 20 depicts the emotion sketch of Karna Parva.

5 Conclusion

The paper has discussed the different techniques used to analyze intricate events of Mahabharat and present them in a lucid and interesting manner to a user without prior knowledge of the text . Various entities present in the Mahabharat text were identified using a Conditional Random Field model



```
emotion_sketches(list_of_emos,contents)

When Sanjaya narrated about the death of Dronacharya to Dhritarashtra, he was very grieved at the defeat of both Bhishma and Drona. He was distresse
Karna devised a new strategy for the Kaurava army 16th day onwards. Even the Pandavas realized that the Kauravas were left with only one warrior, no
One of the major face off in this Parva is the fight between Arjuna and Ashwatthama. He challenged Arjuna and attacked with arrows and divine weapon
Another great battle in this Parva is the face-off between Karna and Nakula. Nakula fought Karna with grit and determination, though Karna was a far
The war between the two armies had become even more intense and terrific after Karna took over the reins of the Kaurava army. Another major fight de
Duryodhana even used his special weapon Sakti to destroy Yudhishthira, but he countered it easily. Thereafter, Yudhishthira attacked him with a mesh
Even under Karna’s leadership, the Pandavas army, especially Arjuna destroyed everyone with his arrows. By the end of the 16th day, the Pandavas fel
The 17th day of the war begun with Bhima’s dominance over the Kauravas army. Bhima even dominated in his battle against Karna, so much so that Karna
The second last day of the war, was packed with close battles across the armies. Bhima defeated Dussasana after engaging in a fierce battle with him
Another of the major fights on that day was the war between Karna’s son, Vrushasena and the Pandava brothers. Vruhasena attacked all of the brothers
Thereafter, Arjuna and Karna engaged in the fiercest of battles in Hindu mythology. The Epic battle is still the most talked about fights in the his
```

Figure 20: Emotion Sketch of Karna Parva

after a comparative analysis. Once the entities were identified, the observations and inferences based on their count, frequency distribution and interactions have been recorded. Various events in the text including Swayamvara and War have been analysed using summary generation models and question answering models. The character analysis provides a first hand impression of the character under consideration and the trails and tribulations which the character has gone through. Emotion analysis draws the flow of emotions and reactions of events described in Mahabharat to be presented in a concise manner to the user. Interested readers can utilize the obtained results from this paper as an incentive for any additional work.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

A. Akbik, Tanja Bergmann, Duncan A. J. Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. Flair: An easy-to-use framework for state-of-the-art nlp. In *NAACL*.

Adnan Akhundov, Dietrich Trautmann, and Georg Groh. 2018. Sequence labeling: A practical approach.

Debarati Das, Bhaskarjyoti Das, and Kavi Mahesh. 2016. A computational analysis of Mahabharata. In *Proceedings of the 13th International Conference on Natural Language Processing*, pages 219–228, Varanasi, India. NLP Association of India.

Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan S. Cowen, Gaurav Nemade, and Sujith Ravi. 2020. Goemotions: A dataset of fine-grained emotions. *CoRR*, abs/2005.00547.

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.

Bhuwan Dhingra, Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2017. Linguistic knowledge as memory for recurrent neural networks. *CoRR*, abs/1703.02620.

Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature*, 585:357–362.

J. D. Hunter. 2007. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

J. Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2022. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 34:50–70.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. *CoRR*, cs.CL/0205028.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CONLL)*.

The pandas development team. 2020. pandas-dev/pandas: Pandas.

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.

Yanyao Shen, Hyokun Yun, Zachary Lipton, Yakov Kronrod, and Animashree Anandkumar. 2017. Deep active learning for named entity recognition. In

Proceedings of the 2nd Workshop on Representation Learning for NLP, pages 252–256, Vancouver, Canada. Association for Computational Linguistics.

Charles Sutton and Andrew McCallum. 2010. An introduction to conditional random fields.

DeepADA: An Attention-Based Deep Learning Framework for Augmenting Imbalanced Textual Datasets

Amit Kumar Sah
Department of Computer Science
South Asian University
New Delhi, India
amitcsrs@students.sau.ac.in

Muhammad Abulaish
Department of Computer Science
South Asian University
New Delhi, India
abulaish@sau.ac.in

Abstract

In this paper, we present an attention-based deep learning framework, DeepADA, which uses data augmentation to address the class imbalance problem in textual datasets. The proposed framework carries out the following functions: (i) using MPNET-based embeddings to extract keywords out of documents from the minority class, (ii) making use of a CNN-BiLSTM architecture with parallel attention to learn the important contextual words associated with the minority class documents' keywords and provide them with word-level characteristics derived from their statistical and semantic features, (iii) using MPNET, replacing the key contextual terms derived from the oversampled documents that match to a keyword with the contextual term that best fits the context, and finally (iv) oversampling the minority class dataset to produce a balanced dataset. Using a 2-layer stacked BiLSTM classifier, we assess the efficacy of the proposed framework using the original and oversampled versions of three Amazon's reviews datasets. We contrast the proposed data augmentation approach with two state-of-the-art text data augmentation methods. The experimental results reveal that our method produces an oversampled dataset that is more useful and helps the classifier perform better than the other two state-of-the-art methods. Nevertheless, we discover that the oversampled datasets outperformed their original ones by a wide margin.

1 Introduction

Textual datasets from different domains generally suffer from the issue of class imbalance, where instances from the majority class outnumber the instances from the minority class by a huge margin. In such a case, the classifier cannot perform well on the minority class dataset; as a result, the minority class instances go undetected. Most research works handle this issue by simply using a random oversampling algorithm without injecting additional knowledge into the minority class dataset. As a result, the resulting model is highly susceptible to overfitting. Nevertheless, a data augmentation-based minority class oversampling approach to handle a class imbalance in a textual dataset has not been well investigated.

Generally, the augmentation techniques generate undesirable documents that do not share the similar distribution of syntax, semantics, and pragmatics of the original data. When it comes to text data augmentation, class-indicating words (keywords) play a significant role (Abulaish and Sah, 2019). We must selectively augment the text, considering the significance of different words. In this process - (i) we should be able to identify the important class-indicating words so that the newly generated document semantically revolves around the class of the original document, (ii) we should be able to characterize the important words that aid the identification class indicating words and ultimately improves the generalization ability.

Deep learning models have emerged as effective classification models and are successful in many domains (Krizhevsky et al., 2012; Huang et al., 2021; Fazil et al., 2021), and have robust pattern learning ability and are widely successful for classification tasks. This paper presents a deep learning-based text data augmentation approach that exploits different deep learning techniques to create a balanced dataset by augmenting the newly generated documents to the minority class dataset to improve the detection efficacy of the classification algorithms on the minority class. The proposed approach first identifies keywords from the minority class data points utilizing a ranking-based weighted approach. Using an attention mechanism, it exploits the identified keywords to extract important contextual words from minority-class documents. In this process, it recognizes the word roles using statistical correlation to measure word occurrence frequencies respective to text categories and semantic similarity to measure word semantics respective to text categories, which helps to find the words semantically similar to text labels. Finally, it utilizes important contextual words to enrich the minority class dataset. The proposed approach oversamples the minority class dataset by generating new documents based on important contextual words and augmenting them to the minority class dataset. The proposed approach seems interpretable and improves the performance of the deep learning classifiers over the augmented datasets.

The rest of the paper is organized as follows. Section 2 presents a brief review of the existing literature on text data augmentation. Section 3 presents a detailed description of the proposed attention-based text data augmentation approach. Section 4 presents the experimental setup and evaluation results. It also presents a comparative analysis of the proposed approach with two state-of-the-art text data augmentation approaches. Finally, section 5 concludes the paper with future research directions.

2 Related Works

Researchers have come up with many approaches including words substitution-based (Wei and Zou, 2019; Kobayashi, 2018; Wu et al., 2019; Abulaish and Sah, 2019), paraphrasing-based (Sennrich et al., 2016; Edunov et al., 2018), and text generation-based (Anaby-Tavor et al., 2020; Liu et al., 2020) solutions.

In (Wei and Zou, 2019), authors introduced Easy Data Augmentation (EDA), a widely used word-replacement based data augmentation method with four basic randomization operations – (i) replacement, (ii) insertion, (iii) swap, and (iv) deletion. They have visualized that these simple operations can improve a classifier's performance on text classification tasks. In (Kobayashi, 2018), authors proposed contextual augmentation for labeled sentences to predict words from a wide range of substitute words, learned using a label-conditional bidirectional language model. In (Abulaish and Sah, 2019), authors showed that augmenting $n-$grams from a minority class document to itself that includes minority class keywords using Latent Dirichlet Allocation (LDA), if any, in the document, can improve the CNN's ability to identify the minority class instances. In (Wu et al., 2019), authors proposed to identify substitute words according to their context, which, apart from the similar meaning, also cares whether the candidates fit in the surrounding context and labels. In (Miao et al., 2020), authors exploited data augmentation to automatically create more labeled training data to fine-tune a language model to derive each aspect-opinion pair's sentiment. In (McCoy et al., 2019), authors insisted on creating new linguistic patterns for text data augmentation using large pre-trained language models.

3 Proposed Attention-Based Deep Learning Framework

In this section, we discuss the proposed attention-based deep learning framework, `DeepADA`, for data augmentation to improve the performance of classifiers on imbalanced text datasets. We evaluate the proposed approach over 3 Amazon reviews dataset, with statistics as shown in Table 1. Table 1 clearly shows that the Amazon reviews dataset suffers from class imbalance problem, as the number of positive reviews is significantly higher than the number of negative reviews. With this inference, we label the positive reviews dataset as the majority class dataset (D_{maj}) and the negative reviews dataset as the minority class dataset (D_{min}). The `DeepADA` performs the following functions:

(i) Extracts keywords out of documents from the minority class, using embeddings generated by a transformer-based language model, as discussed in section 3.1.

(ii) Makes use of a CNN-BiLSTM architecture with parallel attention to learn the important contextual words associated with the minority class documents' keywords, as discussed in section 3.2.

(iii) Using a transformer-based language model, replaces the key contextual terms derived from the oversampled documents that match to a keyword with the contextual term that best fits the context, as discussed in section 3.3.

(iv) Finally, oversamples the minority class dataset to produce a balanced dataset, as discussed in section 3.4.

3.1 Similarity-Based Weighted Keywords Extraction

This section discusses the keyword extraction mechanism that we use in our proposed approach. In order to extract keywords, we use MPNET

(Song et al., 2020), a semantically and contextually robust word embedding technique. MPNET is a transformer-based language model designed to capture the meaning of words, phrases, and documents by encoding them to vectors by inheriting the advantage of both masked language modeling (MLM), adopted in BERT, and permuted language modeling (PLM), adopted in XLNet. In order to generate the embeddings, we use SBERT (Reimers and Gurevych, 2019), which uses siamese and triplet network structures and has proven to be a successful bi-encoder model for generating semantically meaningful sentence embeddings that we can utilize for textual similarity comparisons using cosine similarity. We extract keywords from the minority class dataset (D_{min}) to employ them for their corresponding important contextual words extraction, which we utilize to generate new minority class documents.

In order to extract keywords from the minority class dataset, we first encode each j^{th} document d_j in the minority class dataset (D_{min}) to its embedding vector using SBERT, to extract semantically more meaningful sentence-level embeddings. We consider MPNET as a pre-trained model for SBERT. Then, we encode each i^{th} word w_i from the minority class vocabulary (Vb_{min}) to its embedding vector using MPNET. Thereafter, we calculate cosine similarity ($CSVal$) between each i^{th} word, $w_i \in Vb_{min}$ and j^{th} document $d_j \in D_{min}$, to give $CSVal(w_i, d_j)$ as given by equation 1; where $\overrightarrow{w_i}$ is the embedding vector corresponding to w_i, $\overrightarrow{d_j}$ is the embedding vector corresponding to d_j, and $CSVal(w_i, d_j) \in [-1, 1]$.

$$CSVal(w_i, d_j) = \frac{\overrightarrow{w_i} \cdot \overrightarrow{d_j}}{\|\overrightarrow{w_i}\| \|\overrightarrow{d_j}\|} \tag{1}$$

Once we have calculated $CSVal$ of each word from Vb_{min} with each document in D_{min} comparing their embeddings, we sort them in order such that the words in Vb_{min} which share higher $CSVal$ with most of the documents in D_{min} is ranked higher. In order to achieve this, we assign a score to each word based on its rank. The word score (WS) of word $w \in Vb_{min}$, $WS(w)$ is given by equation 2; where $r(w, d_i)$ is the rank of word w corresponding to i^{th} document $\in D_{min}$, n is the number of documents in D_{min}, and $\mathcal{P}$ is a variable vital in assigning score to a word w based on its rank values corresponding to documents $\in D_{min}$, and it basically penalizes more if the rank corresponding to a document is less; here, we choose $\mathcal{P} = 10$ over other values based on experimental fine-tuning.

$$WS(w) = \sum_{i=1}^{n} r(w, d_i) \times \mathcal{P}^{r(w,d_i)-1} \tag{2}$$

We select only the top k words from Vb_{min} based on their WS value as the minority class keywords (K_{min}).

3.2 Important Contextual Words Extraction from Minority Class Documents

In order to generate additional documents in order to augment it to the minority class D_{min}, we identify the important contextual words corresponding to each minority class document containing the keyword(s). Towards this, we first create a labeled dataset based on the presence of keyword(s) in the minority class documents as discussed in section 3.2.1. We then extract additional word-level features of each word $w \in Vb_{min}$ based on its semantic and statistical property as discussed in section 3.2.2. After that, we learn the important contextual words using a parallel attention-based CNN-BiLSTM model corresponding to the keywords from the keywords-based labeled dataset enriched with the word-level features as discussed in section 3.2. Figure 1 illustrates the important contextual words extraction process from minority class documents.

3.2.1 Keywords-Based Labeled Dataset Creation

In this section, we discuss the creation of a binary labeled dataset for important contextual words extraction D_{icwe} from the minority class dataset D_{min}. We aim to extract important words that aid the classifier most in classifying the keywords. To this end, for each keyword $kw \in K_{min}$, starting from the keywords with the highest $CSVal$ in descending order, we check if a document $r \in D_{min}$, oversample it corresponding to each word $w \in r$. We label class K to the oversampled document if $w = kw$ and assign it to D^{K}_{icwe}, and class NK to the oversampled document if $w \neq kw$ and assign it to D^{NK}_{icwe}. We continue this process until the total number of documents in the minority class dataset, and important words extraction dataset combined is equal to the number of documents in the majority class dataset, i.e., $|D_{min}| + |D^{K}_{icwe}| = |D_{maj}|$.

3.2.2 Word-Level Feature Extraction

In this section, we present the two effective word-level features that depict a word's association with different classes of the dataset based on its statistical and semantic properties. Since we aim to identify the important contextual words corresponding to each $kw \in K_{min}$, we aim to develop a robust classification framework that identifies accurate important contextual words.

(i) **Class Correlation:** This measures how frequently the word co-occurs with the different classes. If the words' frequency is higher in the K class, then its class correlation value is higher for the K class than for the NK class. The class correlation value of a word w corresponding to the class K, $CC(w, K)$ is nothing but weighted log-likelihood ratio, and is given by equation 3; where $p(\frac{w}{K})$ is the probability of observing word w in class c while $p(\frac{w}{K^\complement})$ is the probability of observing word w in class other than K.

$$CC(w, K) = p(\frac{w}{K}) \times log(\frac{p(\frac{w}{K})}{p(\frac{w}{K^\complement})}) \quad (3)$$

Here, for each word w, we calculate $CC(w, K)$ and $CC(w, NK)$ corresponding to the classes K and NK, respectively.

(ii) **Semantic Similarity:** This measures how much semantics the word w share with the label of a class. We take the semantic similarity of a word w with keyword class K as the cosine similarity value of word w with the semantic score of class K, i.e., $CSVal(w, SS(K))$, similar to the calculation in equation 1, and $SS(K)$ as the average of the word vectors of top 100 words $\in K_{min}$ in order of their WS value. We take the top 100 keywords from respective classes to calculate the classwise semantic scores since they represent the better semantic space being the centroid point of top keywords extracted using a more semantically sound approach. Here, for each word w, we calcuate $CSVal(w, SS(K))$ and $CSVal(w, SS(NK))$ corresponding to the classes K and NK respectively.

In this context, providing word-level features to the documents in D_{icwe} will help identify the important contextual words more effectively by exploiting their statistical and semantic property, which depicts their alignment to a particular category.

3.2.3 Keywords-Specific Important Contextual Words Extraction

In this section, we discuss how we extract important contextual words corresponding to the minority class keywords (K_{min}). As from section 3.2.1, we already know that each document d in D_{icwe} is created corresponding to a target word $w_t \in d$ and labeled class K or NK based on whether w_t is in K_{min} or not. After that, in section 3.2.2, we identified two word-level features based on their statistical and semantical properties. In this section, we discuss how we identify the top contextual words that help classify the target word w_t corresponding to which a document $d \in D_{icwe}$ has been created, using a parallel attention-based CNN-BiLSTM model, to ultimately identify the top contextual words corresponding to document $d \in D^{NK}_{icwe}$. We choose the CNN-BiLSTM model to exploit the benefit of both the CNN's feature extraction ability along with the BiLSTM's ability to learn long-term in textual documents (Liu and Guo, 2019; Rhanoui et al., 2019).

Let us suppose d_i is the i^{th} document, and $d_i \in D_{icwe}$ such that $d_i = \{w_1, \dots, w_t, \dots, w_n\}$; w_t and n being the target word and the number of words in the document respectively. `DeepADA` aims to learn the importance of each contextual word $w \in d_i$ while training the model on d_i with emphasis on w_t, where w_t is a target word corresponding to which $d_i \in D_{icwe}$ has been generated and labeled, as discussed in section 3.2. To this end, we have two parallel attention-based CNN, followed by 2 layers stacked BiLSTM, one encoding the preceding context of the document (ENC_p), and the other the following context of the target word (ENC_f) given by equations 4 and 5 respectively.

$$h^p_{w_t} = ENC_p(w_t, h^p_{w_{t-1}}) \quad (4)$$

$$h^f_{w_t} = ENC_f(w_t, h^f_{w_{t-1}}) \quad (5)$$

where ENC_p and ENC_f are two employed CNN-BiLSTM that model the preceding and following context of the target word independently.

With the help of the attention mechanism, variable weights are assigned to all words from the beginning of the document to the target word (encoded by ENC_p), and from the target words towards the end of the document (encoded by ENC_f), depending on their contextual importance.

For encoded vector V_{d_i} of document $d_i \in D_{icwe}$; if hidden state representation of a target word $w_t \in V_{d_i}$ given by the attention-based CNN-BiLSTM classifier is h_{w_t} , then it is passed to a dense-layer to learn its hidden representation h'_{w_t} , as given by equation 6, where W and B represent the weight and bias, respectively. Thereafter, similarity is calculated between h_{w_t} and a vertex vector v_{w_t} which represents the importance of $w_t \in V_{d_i}$. We compute the normalized importance score of w_t using equation 7. The feature-level context vector v_{w_t} is randomly initialized and jointly learned during the training process. Finally, the attention-aware representation of the document d_i is learned and represented as $\mathcal{A}$. It is computed as a weighted sum of the hidden representation of each word, as given by equation 8.

$$h'_{w_t} = tanh(Wh_{w_t} + B) \tag{6}$$

$$\alpha_{w_t} = \frac{\exp(h'_{w_t} v_{w_t})}{\sum_w \exp(h'_{w_t} v_{w_t})} \tag{7}$$

$$\mathcal{A}_{d_i} = \sum_w \alpha_{w_t} h_{w_t} \tag{8}$$

Both ENC_p and ENC_f pass through the processes in equations 6, 7, and 8 simultaneously. The attention-based representation corresponding to ENC_p and ENC_f for document d_i are represented as $\mathcal{A}^p_{d_i}$ and $\mathcal{A}^f_{d_i}$. After we get the attention-based representation vector of the current word in both directions ($\mathcal{A}^p_{d_i}$ and $\mathcal{A}^f_{d_i}$), then we concatenate these two vectors to generate the final representation vector of the document d_i, pass it through a dense layer with 1024 neurons and finally through a softmax layer with 2 neurons. This is done in order to make the model learn and be able to identify the target word given the attention-based weight distribution of the contextual words.

We train the parallel attention-based CNN-BiLSTM model on D_{icwe} dataset. Once we have trained the model, we extract the attention-based vectors $\mathcal{A}^p$ and $\mathcal{A}^f$. These vectors are the attention scores corresponding to words on both sides of the target word w_t. We rank the top words on both sides of w_t based on their attention scores. In this work, we have selected the top 15% words corresponding to both the ENC_p and ENC_f.

3.3 Language Model-Based Documents' Transformation

In this section, we discuss the process of language model-based transformation of documents in D^k_{icwe}. We aim to transform a document $r \in D^k_{icwe}$ to r_t such that the transformed document r_t is a non-duplicate version of r, ensuring that words that are replaced from r to give r_t are contextually similar and gives the semantically similar meaning as r. To this end, we deploy *Fill-Mask* task supported by MPNET, where some of the words in a sentence are masked, and the MPNET model predicts which words best replaces the current word, also known as *mask language modeling*.

We replace the top k important words from each document $r \in D^k_{icwe}$, based on attention score as discussed in section 3.2. Now, for each i^{th} important word $I_{w_i} \in I_w$ where I_w is the list of important words corresponding to r, we learn its best substitute by masking and passing it through the MPNET model. We mask the words in their order of importance, i.e., attention score, and when we mask a word, the rest of the words remain the same. The MPNET model then gives the best word replacement for I_{w_i} in the form of $I^r_{w_i}$. We then replace I_{w_i} by $I^r_{w_i}$ and repeat this for every important word of the document r. In the end, we have the transformed document r_t where all the words $w \in I_w \cap r$ are replaced by their best contextual and semantically similar words given by the MPNET model.

3.4 Balanced Dataset Creation

In this section, we discuss the oversampling process of the minority class dataset D_{min} such that the number of instances in both classes of the dataset is equal. We already know from section 3.2.1 that D^k_{icwe} has been created such that augmenting it to D_{min} gives the balanced dataset. First, we transform each document $r \in D^k_{icwe}$ to give the transformed document r_t as discussed in section 3.3. Finally, we augment D^k_{icwe} with D_{min} to give oversampled minority class dataset AD_{min}, such that $|D_{maj}| = |AD_{min}|$. So, AD_{min} is the final augmented minority class dataset. We replace D_{min} with AD_{min} to give the oversampled balanced dataset.

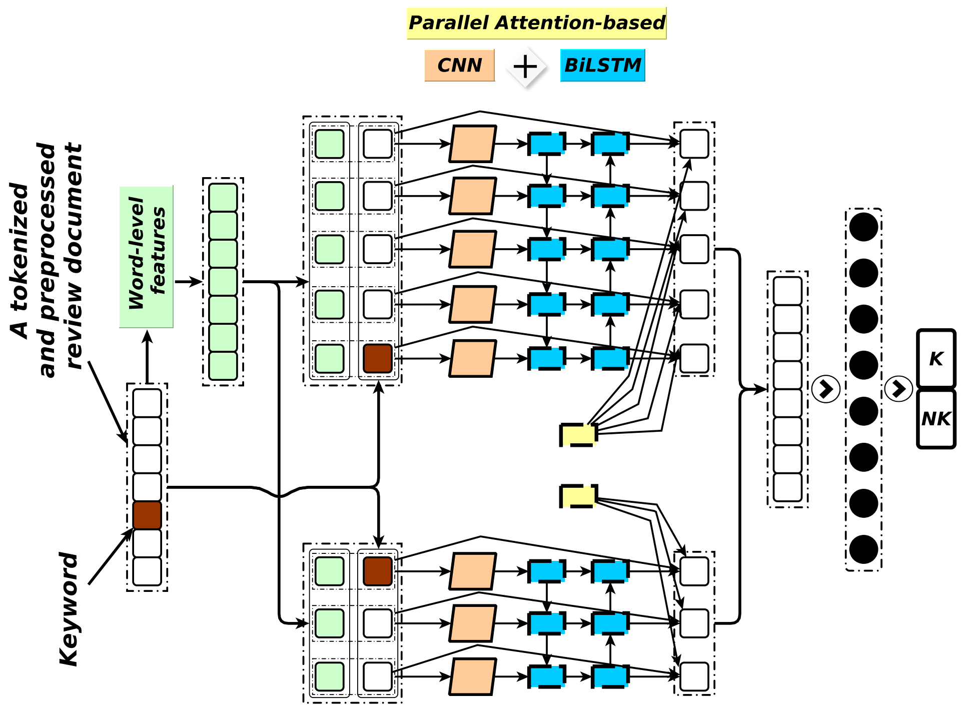


Figure 1: An overview of the important words extraction process

4 Experimental Setup and Results

In this section, we present our experimental setup and discuss the evaluation of the proposed approach. We mention that experiments were performed on a machine with a 2.10 GHz Intel(R) Silver(R) processor and 192G RAM. `DeepADA` was implemented in Keras[1]. For MPNET pre-trained models, we used Transformers[2] library.

4.1 Datasets

We use 3 publicly available Amazon reviews datasets (He and McAuley, 2016) to evaluate `DeepADA` The datasets *musical instruments* (DS_1), *patio lawn and garden* (DS_2), and *automotive* (DS_3) have overall rating on a scale of 1 to 5, 1 being the lowest and 5 being the highest rating provided by the customer, respectively. In this work, we modify this to binary by labeling ratings 1 or 2 as negative reviews and ratings 3, 4, or 5 as positive reviews. The statistics of the modified datasets are shown in Table 1.

Table 1: Statistics of the Amazon review datasets

Dataset	#Reviews	$\#D_{maj}$	$\#D_{min}$	IR
DS_1	10,261	9,794	467	20.97
DS_2	13,272	12,080	1,192	10.13
DS_3	20,473	19,325	1,148	16.83

Review documents differ significantly from standard grammatical structures, and the character limitations compel users to develop creative spellings. Such data needs to be preprocessed more carefully to avoid semantic loss. The preprocessing tasks taken by us are: stop-words, URLs, and hashtag symbols removal, resolving elongated words, emoticons handling, resolving contractions, stemming, and lemmatization.

4.2 Classifier Architecture and Training Details

In this section, we present the classification technique used to validate the effectiveness of `DeepADA`. As discussed, the Amazon reviews datasets are divided into majority class (D_{maj}) and minority class (D_{min}). We used a 2−layer stacked BiLSTM architecture with 256 cells each followed by 2 neurons in the final softmax layer, as we formulated this as a binary classification problem. Other parameters include Xavier Glorot initializer to assign initial weights, adam as an

[1]https://keras.io/
[2]https://huggingface.co/docs/transformers/index

optimizer, dropout to minimize the overfitting effect, with a probability value of 0.2 at the BiLSTM layer, ReLU as an activation function throughout the model, except in the output layer, where we used the softmax function, and $L2$ regularizer with a value of λ as 0.01 over the softmax loss function.

Table 2 gives the statistics of the total number of keywords extracted corresponding to different Amazon reviews datasets to generate the keyword-based labeled dataset, based on the discussion in section 3.2.1. For MPNET-related tasks, we have used the pre-trained model proposed in (Song et al., 2020). For classification tasks throughout this work, we have used $300-$dimensional GloVe embeddings trained on the *Common Crawl* dataset with 840B tokens.

Table 2: Number of keywords extracted corresponding to different Amazon review datasets

Dataset	#Keywords
DS_1	$2,246$
DS_2	$1,172$
DS_3	$2,484$

4.3 Evaluation Metrics

When it comes to the evaluation of imbalanced data, we have very few metrics to consider (Ferri et al., 2009). In the case of a skewed dataset, the usual evaluation metrics, like accuracy, overshadow the performance of the classifier on the minority class. So, we considered reporting the performance of the classification model used in our work only for the minority class and the macro-averaged ones. As evaluation metrics, we considered precision (PR), recall (DR), F$_1$ score (F_1), macro precision ($MacPR$), macro recall ($MacDR$), and macro F$_1$ ($MacF_1$). We chose these evaluation metrics to report the classifier's performance on the minority class and observe whether there is any highly adverse impact on the majority class of the dataset.

4.4 Comparison Approaches and Baseline

In order to establish the efficacy of the proposed model on imbalanced data, we performed a comparative performance evaluation of `DeepADA` with the following two standard text data augmentation techniques:

(1) **EDA – Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks** (Wei and Zou, 2019): In this work, authors have applied 4 simple text data augmentation operations namely – (i) synonym replacement, (ii) random insertion, (iii) random swap, and (iv) random deletion either randomly or regulated by variables in a document. They observed that the classifier's performance was improved on the augmented version of the datasets using these simple data augmentation mechanisms.

(2) **Contextual Augmentation – Data Augmentation by Words with Paradigmatic Relations** (Kobayashi, 2018): In this work, authors have presented a novel text data augmentation technique using different words given by a bi-directional language model and further introduced a label-conditional architecture into the language model. The proposed method produced various words compatibly with the labels of original texts and improved neural classifiers more than synonym-based augmentation. We refer to this work as CDA in the coming sections.

In order to study the effectiveness of incorporating the word-level features in the proposed approach, we created a baseline DeepADA$_b$ by removing word-level features from `DeepADA`.

4.5 Evaluation Results and Comparative Analysis

In order to evaluate `DeepADA`, we balanced the original datasets by oversampling their minority class with new documents generated using `DeepADA`, as discussed in section 3.4. For comparison, we have considered two state-of-the-art text data augmentation techniques, namely EDA (Wei and Zou, 2019) and CDA (Kobayashi, 2018) as well as a baseline `DeepADA`$_b$, which is similar to ablation study that simply excludes the word-level features from `DeepADA`, as discussed in section 4.4. We balanced the original datasets using all EDA, CDA, and `DeepADA`$_b$ for comparison purposes. We performed experimentation on the BiLSTM model discussed in section 4.2 for evaluation purpose. We trained the BilSTM model on 56%, validated it on 14%, and finally tested the model to observe its effectiveness after getting trained on 30% of the datasets. We performed this on both the original and balanced versions of the datasets. While training the BiLSTM, we set the maximum

Table 3: Comparative performance evaluation results of DeepADA on minority class

Approach	DS_1			DS_2			DS_3		
	PR	DR	F_1	PR	DR	F_1	PR	DR	F_1
Original Dataset	45.45	10.42	16.95	37.21	13.48	19.80	35.86	15.03	21.18
EDA (Wei and Zou, 2019)	95.37	98.92	97.11	90.86	97.90	94.25	90.35	98.69	94.33
CDA (Kobayashi, 2018)	95.65	97.15	96.39	**95.98**	94.03	95.00	94.95	97.31	96.12
DeepADA	**97.11**	**99.83**	**98.45**	92.98	98.43	**95.63**	**97.53**	99.41	**98.47**
DeepADA$_b$	96.63	99.66	98.12	92.59	**98.79**	95.59	97.34	**99.57**	98.44

Table 4: Macro comparative performance evaluation results of DeepADA

Approach	DS_1			DS_2			DS_3		
	$MacPR$	$MacDR$	$MacF_1$	$MacPR$	$MacDR$	$MacF_1$	$MacPR$	$MacDR$	$MacF_1$
Original Dataset	70.61	54.90	57.25	64.61	55.62	57.30	65.48	56.71	58.95
EDA (Wei and Zou, 2019)	97.12	97.03	97.04	94.17	93.56	93.71	94.72	95.87	95.16
CDA (Kobayashi, 2018)	96.44	96.46	96.45	95.13	95.09	95.10	96.15	96.15	96.16
DeepADA	**98.47**	**98.40**	**98.42**	**95.65**	**95.48**	**95.48**	**98.47**	**98.45**	**98.45**
DeepADA$_b$	98.14	98.06	98.08	95.64	95.43	95.43	98.45	98.42	98.42

epochs as 100 with the "early stopping" regularization technique to combat overfitting. We have reported the results obtained on test data.

Tables 3 and 4 give the detailed experimental results over minority class datasets and macro-averaged results over all the classes of the datasets, highlighting the performance of DeepADA in comparison to EDA and CDA as well as the baseline approach.

4.5.1 Minority Class Performance

Table 3 shows that the classifier's performance on the original imbalanced datasets was the worst. On the original imbalanced datasets, in terms of F_1 score, we can state that the classifier performance increased with the size of the original dataset. However, when we observed the PR and DR values on the original imbalanced datasets, the same assumption did not stand. It shows that the performance of the deep learning classifier on the original imbalanced datasets was not just abysmal but also noisy. The original imbalanced datasets reported highest PR value of 45.45 on DS_1, and the highest DR and F_1 values of 15.03 and 21.18, both on DS_3.

We were amazed by the classifier's performance on the balanced datasets obtained using any text data augmentation mechanisms. Among the result obtained on the balanced versions of the datasets, the lowest PR, DR and F_1 values were 90.35, 94.03 and 94.25 and were reported on DS_3 using EDA, DS_2 using CDA and DS_3 using EDA respectively; while the highest PR, DR and F_1 values were 97.53, 99.83 and 98.47 and were reported on DS_3, DS_1 and DS_3 all using DeepADA. Overall, DeepADA performed the best in terms of F_1 value. We observed that the performance of DeepADA$_b$ was at par with DeepADA on all evaluation measures over all the datasets. In some cases, like on DS_2 and DS_3, it even outperformed DeepADA in terms of DR value. Out of EDA and CDA, EDA outperformed CDA in terms of DR value on all the datasets, while at the same time, CDA surpasses EDA in terms of PR value on all the datasets. Overall, among EDA and CDA, CDA performed comparatively better than EDA in terms of F_1 value on all datasets except on DS_1. After studying Table 3 in detail, we can conclude that performance over the balanced version of all the datasets created using DeepADA was the best, while the baseline DeepADA$_b$ performed second.

4.5.2 Macro-Averaged Performance

Table 4 shows that the classifier's macro averaged performance on both the classes of the original imbalanced datasets was worst, similar to that observed on the minority class. The original imbalanced datasets reported highest PR value of 70.61 on DS_1, and the highest DR and F_1 values of 56.71 and 58.95, both on DS_3. Compared to the performance on the minority class, the higher value of these evaluation measures signifies that the model can perform well on the majority class but fails miserably to identify the minority class instances correctly.

Table 4 shows that similar to the performance reported on the minority class, the classifier's performance on the balanced datasets obtained using any text data augmentation mechanisms outperforms its performance on the original imbalanced datasets. The detailed study in Table 4 reveals that over all the datasets, the performance of `DeepADA` and `DeepADA`$_b$ over all the macro-averaged evaluation metrics were reported to be first and second best, respectively. Out of EDA and CDA, CDA outperformed EDA in terms of all the evaluation metrics over all the datasets except DS_1. It suggests that both the `DeepADA` and `DeepADA`$_b$ generate high-quality minority class documents, which, when augmented to the minority class dataset, gives a balanced dataset capable of making the classifier perform better on the minority class dataset without degrading its performance on the majority class.

5 Conclusion and Future Work

This paper presents a deep learning-based text data augmentation approach, `DeepADA`, to address the class imbalance issue of classifying textual datasets. The oversampled dataset generated using `DeepADA` can be helpful for deep learning models that extract patterns from the data. Experiments on different datasets show that `DeepADA` significantly outperforms the state-of-the-art methods. The ablation study in the form of the baseline `DeepADA`$_b$ reveals that statistical correlation and semantic similarity are essential for effective word selection. The performance observed on the minority class dataset, and the macro-averaged performance over the 3 datasets indicates that the classifier acquires stronger generalization ability when trained on oversampled datasets generated using `DeepADA`. Exploring more word-level features and extensive study on various transformer-based language models to generate more qualitative oversampled datasets seems a promising future direction of research.

References

Muhammad Abulaish and Amit Kumar Sah. 2019. A text data augmentation approach for improving the performance of cnn. In *11th International Conference on Communication Systems Networks (COMSNETS), Bangalore, India*, pages 625–630. IEEE.

Ateret Anaby-Tavor, Boaz Carmeli, Esther Goldbraich, Amir Kantor, George Kour, Segev Shlomov, Naama Tepper, and Naama Zwerdling. 2020. Do not have enough data? deep learning to the rescue! In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7383–7390.

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.

Mohd Fazil, Amit Kumar Sah, and Muhammad Abulaish. 2021. DeepSBD: A deep neural network model with attention mechanism for socialbot detection. *IEEE Transactions on Information Forensics and Security*, 16:4211–4223.

C. Ferri, J. Hernández-Orallo, and R. Modroiu. 2009. An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30(1):27–38.

Ruining He and Julian McAuley. 2016. Ups and Downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, WWW, page 507–517, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Faliang Huang, Xuelong Li, Changan Yuan, Shichao Zhang, Jilian Zhang, and Shaojie Qiao. 2021. Attention-emotion-enhanced convolutional lstm for sentiment analysis. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–14.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), Louisiana, USA*, pages 452–457.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems, Nevada, USA*, volume 25. Curran Associates, Inc.

Gang Liu and Jiabao Guo. 2019. Bidirectional lstm with attention mechanism and convolutional layer for text classification. *Neurocomputing*, 337:325–338.

Ruibo Liu, Guangxuan Xu, Chenyan Jia, Weicheng Ma, Lili Wang, and Soroush Vosoughi. 2020. Data boost: Text data augmentation through reinforcement learning guided conditional generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9031–9041.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings* *of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy*, pages 3428–3448.

Zhengjie Miao, Yuliang Li, Xiaolan Wang, and Wang-Chiew Tan. 2020. Snippext: Semi-supervised opinion mining with augmented data. In *Proceedings of The Web Conference 2020*, pages 617–628.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese bert-networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China*, pages 3982–3992.

Maryem Rhanoui, Mounia Mikram, Siham Yousfi, and Soukaina Barzali. 2019. A cnn-bilstm model for document-level sentiment analysis. *Machine Learning and Knowledge Extraction*, 1(3):832–847.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *54th Annual Meeting of the Association for Computational Linguistics*, pages 86–96. Association for Computational Linguistics (ACL).

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. MPNet: Masked and permuted pre-training for language understanding. In *Advances in Neural Information Processing Systems*, volume 33, pages 16857–16867. Curran Associates, Inc.

Jason 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), Hong Kong, China*, pages 6382–6388.

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.

Compact Residual Learning with Frequency-Based Non-Square Kernels for Small Footprint Keyword Spotting

Muhammad Abulaish, ***SMIEEE***
Department of Computer Science
South Asian University
New Delhi, India
abulaish@sau.ac.in

Rahul Gulia
Department of Computer Science
South Asian University
New Delh, India
rahulgg619@gmail.com

Abstract

Enabling voice assistants on small embedded devices requires a keyword spotter with a smaller model size and adequate accuracy. It becomes difficult to achieve a reasonable trade-off between a small footprint and high accuracy. Recent studies have demonstrated that convolution neural networks are also effective in the audio domain. In this paper, taking into account the nature of spectrograms, we propose a compact `ResNet` architecture that uses frequency-based non-square kernels to extract the maximum number of timbral features for keyword spotting. The proposed architecture is approximately three-and-a-half times smaller than a comparable architecture with conventional square kernels. On the Google's speech command dataset v1, it outperforms both Google's convolution neural networks and the equivalent `ResNet` architecture with square kernels. By implementing non-square kernels for spectrogram-related data, we can achieve a significant increase in accuracy with relatively few parameters, as compared to the conventional square kernels that are the default choice for every problem.

1 Introduction

Keyword detection systems (KWS) are implemented on embedded or mobile devices to detect predefined keywords in an audio stream. These words can function as *wake words* or *trigger words* for intelligent voice assistants (e.g., *Hey Siri*, *Alexa*, or *Okay Google*) or as simple speech commands (e.g., *yes*, *no*, *on*, *stop*, etc.). Due to the nature of deployment, these systems must have a reasonable compromise between a small footprint and high accuracy. However, implementing a fast, compact, and highly accurate KWS model that can be deployed on embedded or mobile devices with limited hardware and computation is a significant challenge.

Recent studies have demonstrated that convolution neural networks (CNNs) perform well in the audio domain as well. CNNs are predominantly utilized for image-related tasks. CNN's lowest layers typically learn to detect edges. These edges can be oriented in any way. However, one cannot predict which kernel will acquire a given feature. Also, since filter dimensions have a spatial meaning, square kernels are widely used to preserve symmetry. It is common practice in the audio domain to transform an audio stream into a spectrogram in which the X-axis represents time and the Y-axis represents frequency. Then, these spectrograms are fed to two-dimensional CNNs as input.

Spectrograms are a visual representation of the audio intensity over time at various frequencies present in a specific waveform. On spectrograms, the X-axis represents time and the Y-axis represents frequency. The images are then sent to CNN, which performs the feature extractions. The composition of spectrograms is known in advance, i.e., time is represented on the X-axis and frequency on the Y-axis. In order to capture frequency-based and time-based characteristics, customized kernel designs can be implemented. A number of attempts have been made to use custom rectangular kernels for speech emotion detection and music rhythm classification (Pons et al., 2016; Badshah et al., 2019).

In this paper, we propose a compact `ResNet` architecture, i.e., CNNs with residual learning, that uses frequency-based non-square kernels to capture the maximum number of timbral features for keyword detection. Typically, the timbral features are taller than they are wide. Also, because a 3×1 matrix has fewer parameters than a 3×3 matrix, we experimented with non-square kernels for the KWS use-case, with the goal of reducing the number of parameters while maintaining decent accuracy. As a result, it is worthwhile to employ non-square kernels. Figure 1 depicts an example non-square frequency-based kernel. Our architecture is a modification of the res8 model (Tang and

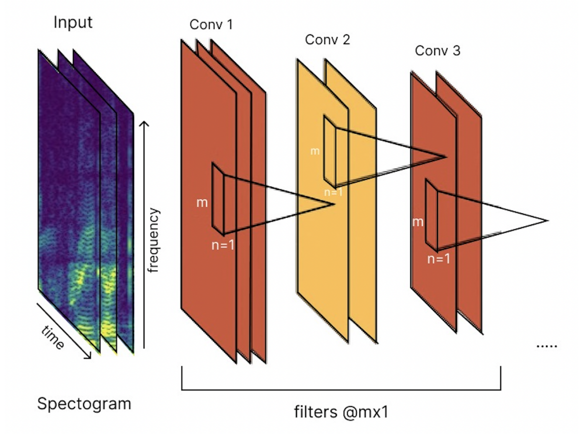


Figure 1: Non-square frequency-based $m \times 1$ kernels

Lin, 2018) with non-square kernels, and it is approximately 3.5 times more compact than the res8. In terms of number of parameters, our architecture also outperforms Google's best CNN (Sainath and Parada, 2015). Compared to the res8 model, the proposed architecture achieves higher evaluation performance and has a smaller footprint.

The remainder of the article is structured as follows. Brief summaries of recent and pertinent keyword spotting research are presented in Section 2. The functional details of our suggested approach for the keyword spotting use-case are presented in Section 3. The experimental setup and results are presented in Section 4. It also provides a comparative analysis of our proposed approach with respect to the two SOTA models, Tang and Lin (2018) and Sainath and Parada (2015), for keyword spotting use-case. Finally, the paper concludes with a discussion of future research directions in Section 5.

2 Related Works

Traditionally, hidden Markov models (HMMs) based approaches were used for KWS (Wilpon et al., 1990; Rose and Paul, 1990). These models were challenging to train, were computationally expensive, and had relatively long latency during inference. Some other techniques used recurrent neural networks (RNNs) like in (Fernández et al., 2007), but they suffered from high latency. After that, Chen et al. (2014) proposed deep neural networks (DNNs) with rectified linear unit (ReLU) activation functions that outperformed HMM-based models with low latency. But the drawback with DNNs is that they ignores the audio's local temporal and spectral correlations. To capture these correlations, variations of CNN-based KWS were introduced . They achieved better results with reduced footprints. Combining the strength of CNNs and RNNs, Arik et al. (2017) experimented with the convolution recurrent neural network-based KWS systems.

In August 2017, the Google's speech command dataset by Warden (2018) was released as a benchmarking dataset for evaluating KWS systems. Warden (2018) also released a baseline model based on the convolution architecture of Sainath and Parada (2015), achieving 85.4% accuracy in v1 version of the dataset. The related Kaggle competition was also organized, where the winner achieved 91% accuracy on the v1 version at that time. Publication of the Google's speech command dataset led to an acceleration in the research. Here, we will discuss the most relevant work as per the paper's objective.

He et al. (2016) (`ResNets`) significantly advance deep learning, and hence they have also been adopted in various audio tasks like automatic speech recognition (Xiong et al., 2018) and speaker identification (Yun et al., 2019). Tang and Lin (2018) further experimented with compact residual architecture using dialated convolution to enlarge the size of the receptive field exponentially with the depth of the network, resulting in improved accuracy. They also experimented with model width by decreasing number of filters. Szegedy et al. (2016) proposed several enhancements to the inception network (Szegedy et al., 2015). They replaced several convolutions with lower dimension convolutions to decrease the parameters. For example, a 7x7 convolution was replaced with four layers by using 1x7 and 7x1 convolutions twice. To reduce model footprint, a number of recent works like time delay neural network (TDNN), attention mechanism, and temporal convolutional network (TCN) are done (Sun et al., 2017; Shan et al., 2018; Choi et al., 2019).

There have been some attempts to use rectangular kernels for speech emotion detection and music rhythm classification (Pons et al., 2016; Badshah et al., 2019). The authors of (Hoogeboom et al., 2018) have used hexagonal kernels that utilized symmetry equivariance and in-variance of images. Our paper uses 2D convolution with non-square kernels ($m \times 1$) convoluted in the frequency domain to capture the maximum amount of timbral features, reducing the computation and number of operations.

This study focuses on the family of CNN models, as they continue to serve as the benchmark for KWS systems. For KWS systems, we desire an analysis between square and non-square kernels. Since CNNs have a simple architecture, we have conducted our experiments with CNN and residual blocks. In order to perform a more accurate benchmarking, we have referred to the results that were published in (Tang and Lin, 2018).

3 Proposed `ResNet` Architecture with non-square kernel

In this section, the functional details of our proposed `ResNet` architecture with non-square kernels are described. Figure 2 provides a visual representation of the architectural layout of the proposed `ResNet` with non-square kernel.

3.1 Feature Extraction and Preprocessing

Table 1: Parameters used for res8-3x1

Type	Height (m)	Width (n)	Filters (N)	#Parameters
conv	9	5	45	6,075
avg-pool	3	4	45	-
res x 6	3	1	45	36.4K
avg-pool	3	4	45	-
softmax	-	-	12	552
Total	-	-	-	43K

Table 2: Parameters used for res8-5x1

Type	Height (m)	Width (n)	Filters (N)	#Parameters
conv	9	5	45	6,075
avg-pool	3	4	45	-
res x 6	5	1	45	60.7K
avg-pool	3	4	45	-
softmax	-	-	12	552
Total	-	-	-	67.3K

Table 3: Parameters used for res8-7x1

Type	Height (m)	Width (n)	Filters (N)	#Parameters
conv	9	5	45	6,075
avg-pool	3	4	45	-
res x 6	7	1	45	85K
avg-pool	3	4	45	-
softmax	-	-	12	552
Total	-	-	-	91.6K

The input audio stream is converted into mel-scaled spectrograms with a length of $FFTwindow$ as 2048 and 512 samples between successive frames. The spectrogram is then converted to decibels (dB), with the highest dB being 80. We use the `Librosa` Python library to perform the conversion. Spectrogram data is then normalized, standardized, and converted to three dimensions by repeating the matrix along all three axes $[X, X, X]$. The spectrogram images are then scaled to 128×64 pixels. Figure 3 illustrates a sample of the generated spectrogram images.

Table 4: Parameters used for res8-9x1

Type	Height (m)	Width (n)	Filters (N)	#Parameters
conv	9	5	45	6,075
avg-pool	3	4	45	-
res x 6	9	1	45	109K
avg-pool	3	4	45	-
softmax	-	-	12	552
Total	-	-	-	115.9K

3.2 Proposed Architecture

Our network architecture contains residual blocks, as described in (He et al., 2016), in which it is proposed that it is simpler to learn residuals:

$$H(x) = F(x) + x$$

than the actual mapping $F(x)$ for a model with greater depth.

Instead of using small squared CNN kernels (e.g., 3×3 or 5×5), we use non-square kernels of $m \times 1$ with varying values of m because such kernels may be able to learn more timbral features than standard square kernels. From an audio standpoint, these kernels are expected to learn fundamental audio features such as frequency, pitch, and timbre, among others. Such kernels may be incapable of learning more about the time axis, i.e., rhythmic or tempo-related features. Taking into account the keyword spotting use-case, however, these kernels reduce the number of parameters and memory footprint, motivating us to move in this direction.

Our first layer is a bias-free $m \times n$ 2D convolution kernel, where m and n are, respectively, the height and width (m = 9, n = 5). Here, n is greater than 1 for the first layer to increase the receptive field, and in subsequent layers, the time axis also contributes. This layer has a stride of 2×2 which helps to reduce the size of the model. After the first layer, we added a 3×4 average pooling layer to reduce the input dimensions (3×4 kernel size yielded better results than 4×3 in our setup). Our residual block is comprised of a bias-free $m \times 1$ 2D convolution kernel, an activation function, and a batch normalization layer. Since the neuron cannot determine its own firing pattern, an activation function is used to determine whether or not a neuron should be activated. Before passing the input to the subsequent layer, the non-linear transformations

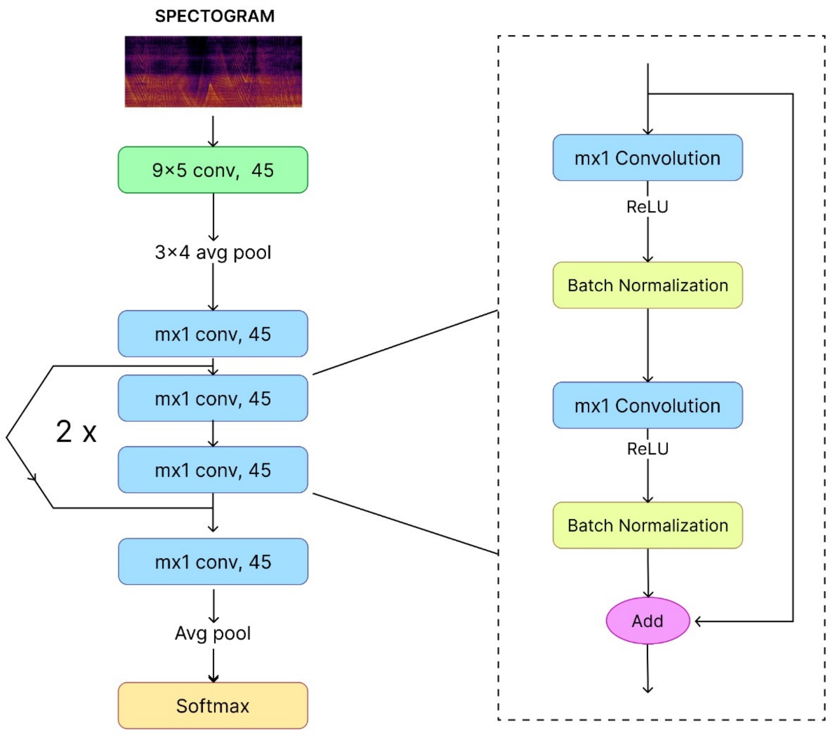


Figure 2: Architecture of the proposed `ResNet` with non-square kernel ($m \times 1$)

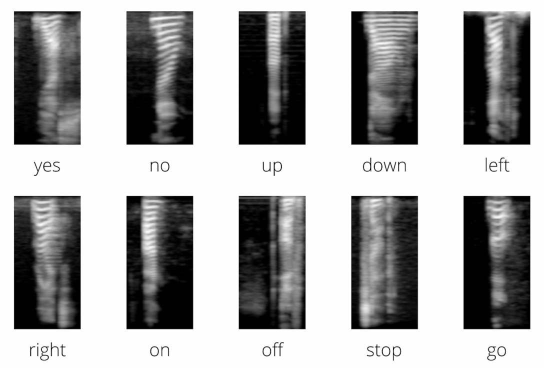


Figure 3: Spectrogram images of different classes

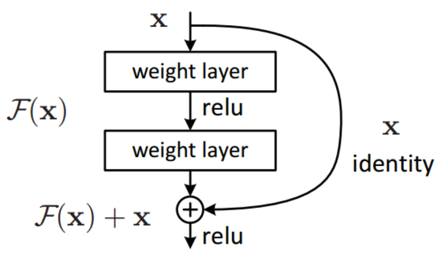

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 4: A residual block proposed in (He et al., 2016)

are applied. We have used the `ReLU` activation function, which outputs the input directly if it is positive and 0 if it is negative. Formally, it is defined as follows:

$$Relu(z) = max(0, z)$$

We added a chain of six residual blocks. In the end, we included batch normalization and a non-residual convolution layer. Every mini-batch's weights are normalized via batch normalization. Consequently, it stabilizes the network and drastically reduces the number of training epochs necessary to train deep neural networks. Following the addition of a linear layer, the output then passes through a `softmax` layer that generates a class probability distribution (see figure 2).

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1, 2, \dots, K$$

$N = 45$ feature maps are utilized across all convolution layers. We tested four `ResNet` variants by varying m in the residual block: res8-3x1 with 43K parameters (Table 1), res8-5x1 with 68K parameters (Table 2), res8-7x1 with 92K parameters (Table 3), and res8-9x1 with 115K parameters (Table 4).

4 Experimental Setup and Results

This section describes the experimental design and results. It also provides a comparative analysis of our proposed method with two SOTA models, Tang and Lin (2018) and Sainath and Parada (2015), for keyword spotting use-case.

4.1 Dataset

We use Google's speech commands dataset v1 by Warden (2018) for training and benchmarking purposes for our proposed network that was released in August 2017 under a Creative Commons license. The dataset includes approximately 64727 one-second long utterances of 30 short words, sampled at 16k Hz and recorded by different individuals. The distribution of words in the dataset is depicted in Table 5.

In accordance with Google's implementation, we categorized the audios into 12 classes – *yes*, *no*, *up*, *down*, *left*, *right*, *on*, *off*, *stop*, *go*, *unknown* (remaining words), and *silence* (no speech detected). The blog post announcing the dataset mentions Google's `TensorFlow` implementation of Sainath and Parada (2015) models, which are used for comparison alongside Residual networks proposed by Tang and Lin (2018). Following the publication, we compared the results of the experiments to the v1 test data. Based on the Warden (2018), the dataset is divided into 80% training set, 10% validation set, and 10% test set. This resulted in approximately 23000 examples for training and 2700 for validation and testing.

Table 5: Word distribution in Google's speech command dataset v1

Word	#Utterances	Word	#Utterances	Word	#Utterances
bed	1,713	house	1,750	sheila	1,734
bird	1,731	left	2,353	six	2,369
cat	1,733	marvin	1,746	stop	2,380
dog	1,746	nine	2,364	three	2,356
down	2,359	no	2,375	tree	1,733
eight	2,352	off	2,357	two	2,373
five	2,357	on	2,367	up	2,375
four	2,372	one	2,370	wow	1,745
go	2,372	right	2,367	yes	2,377
happy	1,742	seven	2,377	zero	2,376

4.2 Evaluation Metrics

The proposed method is evaluated based on its accuracy, which is formally defined using True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) values in the following equation.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

TP is defined in the preceding equation as the test result that correctly indicates the presence of a condition. FP is defined as a test result that incorrectly indicates the presence of a specific condition. The test result that correctly indicates the absence of a condition is defined as TN. Finally, FN is defined as a test result that incorrectly indicates the presence of a specific condition.

In addition to accuracy, we have considered the footprint size of the proposed method in terms of the number of parameters, which is calculated using the following formula:

$$Parameters = (fs * pf + 1) * N$$

In the above equation, *fs* represents the $mtimesn$ dimensions of the kernels used; *pf* represents the number of kernels used in the previous layer; and *N* represents the number of kernels used in the current layer. As the bias term, 1 is added to the previous equation.

4.3 Model Training

Using the `PyTorch` framework, we trained and evaluated various models. We used `AdamW` (Loshchilov and Hutter, 2017) as our optimizer with a learning rate of 3e-4 on a mini-batch of 64 samples with 0.001 weight decay at each layer except *LayerNorm* and *Bias*. For the LR scheduler, *ReduceLROnPlateau* is selected as the learning rate scheduler that reads the metric quantity and reduces the learning rate by a certain factor (0.8 in our case) if no improvement is observed for *patience* number of epochs (*patience* is set to 2 in our configuration). We utilized down-sampling to address class imbalance, if any. To prevent the occurrence of over-fitting, we also utilized early stopping on the validation loss with a *patience* of 5. As our loss function, we employ cross entropy, which is defined as follows, wherein K is the number of classes, y is a binary indicator to check whether the class label c is the correct classification for observation o, and p is the predicted probability of observation o for class c.

$$CrossEntropyLoss = -\sum_{c=1}^{K} y_{o,c} \log(p_{o,c})$$

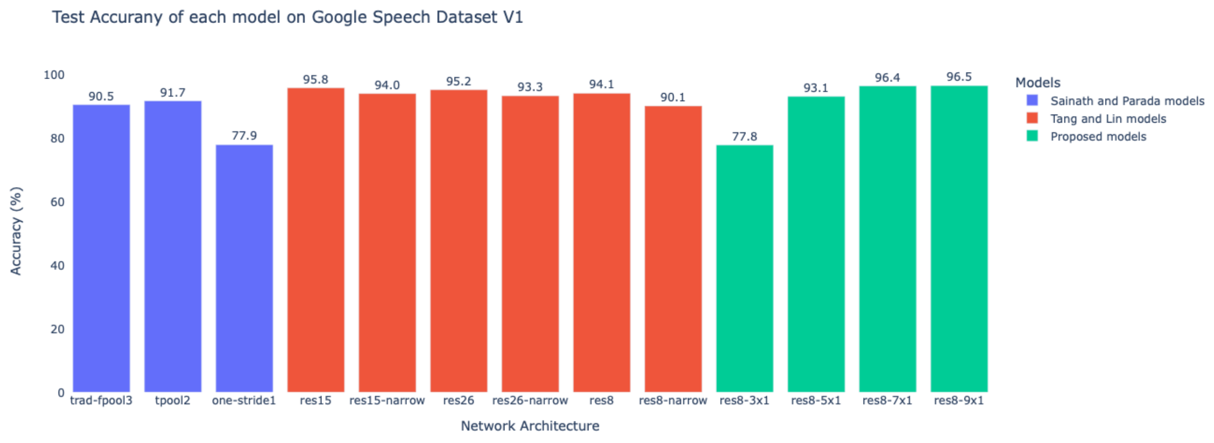


Figure 5: Visualization of accuracy of all models on Google's speech command dataset v1

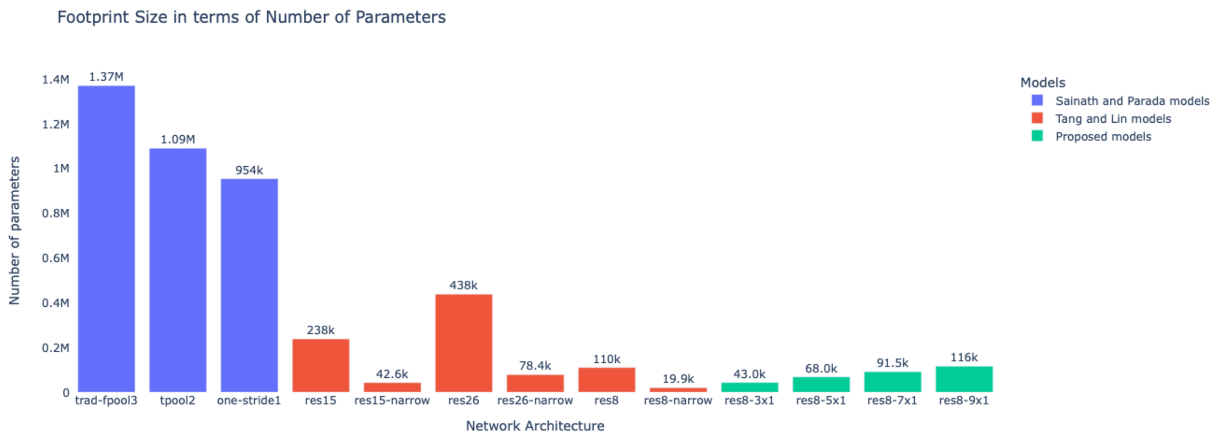


Figure 6: Visualization of footprint size of all models in terms of number of parameters

4.4 Results

For benchmarking, we employ the CNN variants proposed by Sainath and Parada (2015), namely *trad-fpool3*, *tpool2*, and *one-stride1*. In addition, we compared our findings to the compact `ResNet` models proposed in the (Tang and Lin, 2018). In consideration of the small footprint keyword spotting use-case, we employ *no. of parameters* and *model size (MB)* in addition to an evaluation metric for benchmarking purposes. Our proposed method yields comparable results with few parameters. As demonstrated in the table above, as m increases, the model is able to capture more timbral features, resulting in a certain degree of accuracy improvement.

Table 6: Test accuracy of our proposed models in terms of accuracy and number of parameters

Proposed Model	Accuracy	#Parameters
res8-3x1	77.8%	43K
res8-5x1	93.1%	68K
res8-7x1	96.4%	91.6K
res8-9x1	96.5%	115.9K

4.5 Comparative Analysis

In this section, we present the results of a comparative analysis between our proposed `ResNet` model and the Tang and Lin (2018) and Sainath and Parada (2015) models. These models are briefly described in the following paragraphs.

1. Sainath and Parada (2015) models
   – trad-fpool3: It is their base model. It consists of 2 convolution, 1 linear layer,

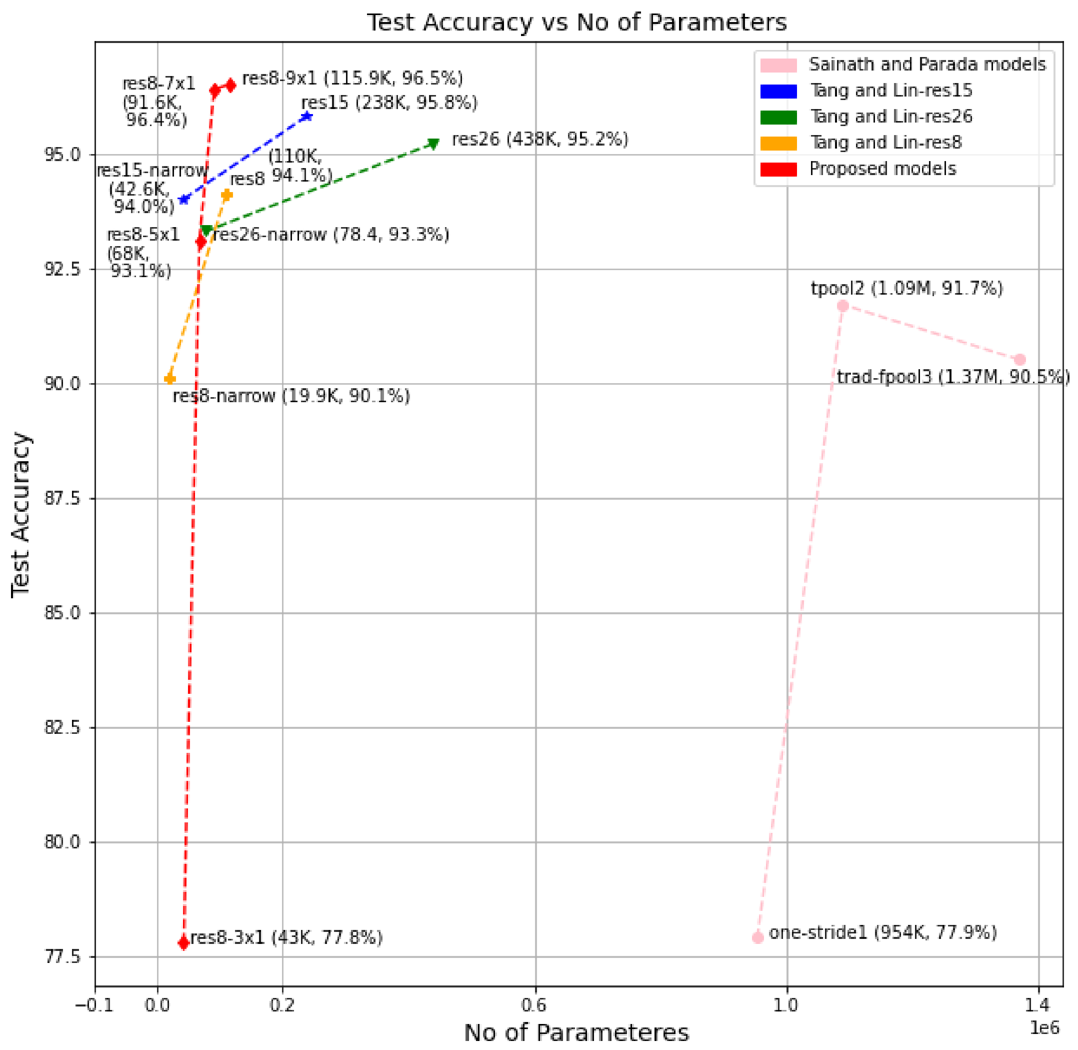


Figure 7: Number of parameters vs. accuracy over Google's speech command dataset v1 (see Table 7)

Table 7: Comparative analysis of our proposed model with other models

Models		Accuracy	#Parameters
Sainath and Parada (2015) models	trad-fpool3	90.5%	1.37M
	tpool2	91.7%	1.09M
	one-stride1	77.9%	954K
Tang and Lin (2018) models	res15	95.8%	238K
	res15-narrow	94.0%	42.6K
	res26	95.2%	438K
	res26-narrow	93.3%	78.4K
	res8	94.1%	110K
	res8-narrow	90.1%	19.9K
Proposed models	res8-3x1	77.8%	43K
	res8-5x1	93.1%	68K
	res8-7x1	96.4%	91.6K
	res8-9x1	96.5%	115.9K

hidden and softmax layer with poolng in frequency axis.

– tpool2: The most accurate variant they explored. It's the variant of the base model with pooling in time axis.

– one-stride1: Their best compact variant is the variant of the base model with stride in frequency axis.

2. Tang and Lin (2018) models

   – res15: ResNet model with 15 layers and 45 kernels

   – res15-narrow: ResNet model with 15 layers and 19 kernels

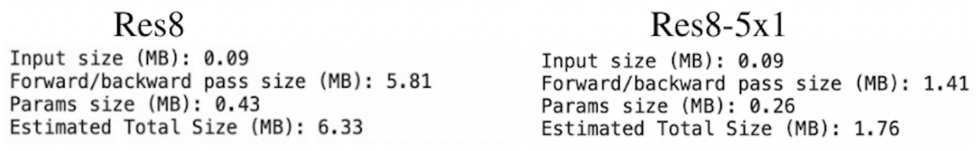
   – res26: ResNet model with 26 layers and 45 kernels

   – res26-narrow: ResNet model with 26 layers and 19 kernels

   – res8: ResNet model with 8 layers and 45 kernels

   – res8-narrow: ResNet model with 8 layers and 19 kernels

Except for res8-3x1, all of our proposed models (see Table 6) outperform Sainath and Parada (2015)


Res8

```
Input size (MB): 0.09
Forward/backward pass size (MB): 5.81
Params size (MB): 0.43
Estimated Total Size (MB): 6.33
```

Res8-5x1

```
Input size (MB): 0.09
Forward/backward pass size (MB): 1.41
Params size (MB): 0.26
Estimated Total Size (MB): 1.76
```

Figure 8: Model size (in terms of MB) comparison between models with square kernel (left) and non-square kernel (right)

in terms of accuracy (Figure 5) and number of parameters (Figure 6), as shown in Table 7. Our res8-5x1 provides slightly less accuracy than the compact `ResNet` models proposed in (Tang and Lin, 2018), but with a much smaller number of parameters.

With only 91.6K parameters, the res8-7x1 model that employs a 7×1 filter size achieves the optimal balance between accuracy and number of parameters, outperforming all models proposed in (Sainath and Parada, 2015) and (Tang and Lin, 2018). Using non-square kernels has been shown to provide adequate and, in some cases, superior accuracy with a small number of audio domain parameters.

Figure 6 illustrates the footprint size in terms of the number of parameters for each model considered for benchmarking. Except for the narrow models by Tang and Lin (2018), all of our proposed models have a smaller number of parameters and comparable accuracy when compared to the other models used for benchmarking. According to their paper, when comparing narrow versus wide models, the number of kernels has a greater effect on accuracy than model depth. Our emphasis is on employing non-square kernels for audio domain in order to reduce model footprint. We employ the same number of kernels ($N = 45$) as the res8 model from (Tang and Lin, 2018).

Compared to models with squared kernels, models with non-squared kernels have a tendency to have a smaller number of parameters with comparable or better accuracy. In addition, we use the same input size for models with a square (res8) kernel and a non-square (res8-5x1) kernel. Figure 8 demonstrates that res8-5x1 is approximately 3.5 times smaller than the res8 model.

5 Conclusion and Future Work

In this paper, we have presented a fast, small footprint model for real-time KWS with non-square kernels ($m \times 1$) that may be useful for small embedded devices. Depending on the characteristics of the spectrogram, non-square kernels may be a suitable alternative to square kernels, provided that non-square kernels have fewer trainable parameters than square kernels. Experiments indicate that non-square kernels reduce model size by reducing the number of required parameters without degrading performance. The frequency-based kernels have few parameters, and the proposed architecture demonstrates satisfactory performance during the evaluation phase. Our proposed work may inspire other researchers and developers to experiment with network architecture based on the dataset and use-case. After comprehending the structure of our data, i.e., spectrogram, we have utilized non-square kernels in the audio domain for KWS. It might be worthwhile to consider conducting additional research on benchmarking different architectures using non-square kernels in the future.

References

Sercan O Arik, Markus Kliegl, Rewon Child, Joel Hestness, Andrew Gibiansky, Chris Fougner, Ryan Prenger, and Adam Coates. 2017. Convolutional recurrent neural networks for small-footprint keyword spotting. *arXiv preprint arXiv:1703.05390*.

Abdul Malik Badshah, Nasir Rahim, Noor Ullah, Jamil Ahmad, Khan Muhammad, Mi Young Lee, Soonil Kwon, and Sung Wook Baik. 2019. Deep features-based speech emotion recognition for smart affective services. *Multimedia Tools and Applications*, 78(5):5571–5589.

Guoguo Chen, Carolina Parada, and Georg Heigold. 2014. Small-footprint keyword spotting using deep neural networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4087–4091. IEEE.

Seungwoo Choi, Seokjun Seo, Beomjun Shin, Hyeongmin Byun, Martin Kersner, Beomsu Kim, Dongyoung Kim, and Sungjoo Ha. 2019. Temporal convolution for real-time keyword spotting on mobile devices. *arXiv preprint arXiv:1904.03814*.

Santiago Fernández, Alex Graves, and Jürgen Schmidhuber. 2007. An application of recurrent neural networks to discriminative keyword spotting. In *International Conference on Artificial Neural Networks*, pages 220–229. Springer.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Emiel Hoogeboom, Jorn WT Peters, Taco S Cohen, and Max Welling. 2018. Hexaconv. *arXiv preprint arXiv:1803.02108*.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Jordi Pons, Thomas Lidy, and Xavier Serra. 2016. Experimenting with musically motivated convolutional neural networks. In *2016 14th International Workshop on Content-Based Multimedia Indexing (CBMI)*, pages 1–6.

Richard C Rose and Douglas B Paul. 1990. A hidden markov model based keyword recognition system. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 129–132. IEEE.

Tara Sainath and Carolina Parada. 2015. Convolutional neural networks for small-footprint keyword spotting. In *Interspeech*.

Changhao Shan, Junbo Zhang, Yujun Wang, and Lei Xie. 2018. Attention-based end-to-end models for small-footprint keyword spotting. *arXiv preprint arXiv:1803.10916*.

Ming Sun, David Snyder, Yixin Gao, Varun K. Nagaraja, Mike Rodehorst, Sankaran Panchapagesan, Nikko Strom, Spyridon Matsoukas, and Shiv Naga Prasad Vitaladevuni. 2017. Compressed time delay neural network for small-footprint keyword spotting. In *INTERSPEECH*.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Raphael Tang and Jimmy Lin. 2018. Deep residual learning for small-footprint keyword spotting. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5484–5488. IEEE.

Pete Warden. 2018. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*.

Jay G Wilpon, Lawrence R Rabiner, C-H Lee, and ER Goldman. 1990. Automatic recognition of keywords in unconstrained speech using hidden markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(11):1870–1878.

Wayne Xiong, Lingfeng Wu, Fil Alleva, Jasha Droppo, Xuedong Huang, and Andreas Stolcke. 2018. The microsoft 2017 conversational speech recognition system. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5934–5938. IEEE.

Sungrack Yun, Janghoon Cho, Jungyun Eum, Wonil Chang, and Kyuwoong Hwang. 2019. An end-to-end text-independent speaker verification framework with a keyword adversarial network. *arXiv preprint arXiv:1908.02612*.

Unsupervised Bengali Text Summarization Using Sentence Embedding and Spectral Clustering

Sohini Roy Chowdhury and **Kamal Sarkar** and **Arka Maji**
jukamal2001@yahoo.com
Department of Computer Science and Engineering
Jadavpur University
188, Raja S.C. Mallick Rd
Kolkata-700032,West Bengal
India

Abstract

Single document extractive text summarization produces a condensed version of a document by extracting salient sentences from the document. Most significant and diverse information can be obtained from a document by breaking it into topical clusters of sentences. The spectral clustering method is useful in text summarization because it does not assume any fixed shape of the clusters, and the number of clusters can automatically be inferred using the Eigen gap method. In our approach, we have used word embedding-based sentence representation and a spectral clustering algorithm to identify various topics covered in a Bengali document and generate an extractive summary by selecting salient sentences from the identified topics. We have compared our developed Bengali summarization system with several baseline extractive summarization systems. The experimental results show that the proposed approach performs better than some baseline Bengali summarization systems it is compared to.

1 Introduction

With the advancement of search engines, we are flooded with information. This information overload problem affects the proficiency of decision-making of humans. Instead of time waste, it also affects the capacity of humans. In today's world where each day technology is changing our daily life, the human brain plays an important role there. So it is unworthy to waste the human brain and time in a negative way. Having a crux with relevant information from a long document manually is a very tedious task. Text summarization is a very useful solution to this information overload problem. Text summarization helps to create a condensed version of a document by selecting sentences with pertinent information from the document. Text summarization demands well understanding of the document to create the gist. Text summarization can be categorized into two types: extractive and abstractive. Extractive summarization aims to generate a summary by selecting textual segments or sentences from the document whereas abstractive summaries are generated from the document by introducing new words or phrase which may not be present in the original document. Although abstractive summaries are more human-like than extractive summaries, the state-of-the art abstractive summarization approaches are not good enough in producing an abstract from a longer document. Many existing abstractive summarization approaches use two step process. in the first step, an extract is generated. and in the next step, an abstract is generated by reformulating the sentences in the extract(Sarkar, 2010). Thus the extractive summarization is useful. Moreover, Bengali is a resource-scarce language and abstractive summarization requires a large amount of language resources which are not available for Bengali language. This motivates us to work on Bengali extractive summarization.

Capturing connectivity among sentences of a document is helpful to group similar sentences and create a condensed extract. Sentence clustering is an unsupervised method that groups similar sentences and produces clusters. Traditional clustering algorithms though widely used earlier have some pitfalls which are overcome using spectral clustering. Spectral clustering emphasizes creating more accurate clusters than traditional clustering algorithms as it does not make assumptions about the shape of the cluster. Spectral clustering utilizes the connectivity of data points. If two data points appear side by side but are not connected, spectral clustering will not group them together. The main benefits of using spectral clustering in document segmentation are that the clusters produced by this method do not follow any fixed shape. We assume that the clusters representing topics are non-Gaussian. We consider that the spectral clustering algorithm is suitable for segmenting a document into multiple topical clusters where a cluster represents a topical segment that consists of semantically similar sentences appearing in close positional proximity.

In our approach, each sentence vector is computed by averaging the word embedding vectors obtained using fasttext [1] open source. After obtaining the sentence vector by averaging the word vectors, the position of the sentence is included as an additional dimension of the sentence vector. Position information is considered to encourage locally coherent sentences to fall in the same cluster. Clusters are ranked according to the average position number of the sentences in the cluster and a summary is created by choosing the most relevant sentences from the ranked clusters. Sentence selection from the cluster is also done in an effective way. The efficacy of our approach lies in the effectiveness of spectral clustering in segmenting the document into multiple topical clusters.

The approach proposed in this paper differs from the existing approaches (Günes and Dragomir R., 2004)(Sarkar, 2009a)(Sarkar, 2009b)(Sarkar, 2008)(Sarkar, 2012a)(Sarkar, 2012b)(Sarkar and Bandyopadhyay, 2005). We use the spectral clustering algorithm to segment a document into multiple topical clusters and create a summary by choosing topic-wise most relevant sentences. On the other hand, the existing approaches (Günes and Dragomir R., 2004)(Sarkar, 2009a)(Sarkar, 2009b)(Sarkar, 2008)(Sarkar, 2012a)(Sarkar, 2012b)(Sarkar and Bandyopadhyay, 2005) decompose the entire document into a collection of sentences and rank the sentences based on some features to create a summary. So, our proposed approach uses an effective clustering-based method that produces a summary covering all important topics in a document.

Our paper is set up in the following manner. Related work is discussed in section 2. Our proposed methodology is explained in section 3. Section 4 highlights the dataset used in the approach results and comparison among existing models. Section 5 concludes the paper.

2 Related Work

In the area of extractive text summarization, the early approaches used various heuristic algorithms to identify important segments from a document. The methods that include features like sentence position, word frequency, and key phrases to extract salient sentences from the document have been presented in (Baxendale, 1958)(Edmundson, 1969)(Luhn, 1958). Most early text summarization algorithms faced the redundancy problem or the diversity problem. So, to deal with these problems and assuring good coverage, clustering of sentences is used (Jain and Dubes, 1988). The idea of employing a clustering algorithm for text summarization was well described in (Sarkar, 2009a). This approach used three steps for text summarization: histogram-based clustering algorithm for sentence clustering, ordering of clusters, and extraction of summary-worthy sentences from the clusters to create the summary.

In (Jing and McKeown, 2000), a hierarchical agglomerative clustering algorithm was used to create clusters of sentences. To create a summary, sentences were chosen in order from largest to smallest cluster. Another clustering-based approach presented in (Wan and Yang, 2008) incorporates cluster-level information in a graph model for ranking sentences.

However, the early works (Sarkar, 2009a) suggest that the performance of clustering-based text summarization heavily depends on the quality of clusters produced. Clustering algorithms perform well when we have a clear idea regarding attributes of data points (Jin, 2006). Clustering based on compactness highlights spatial proximity among data points. For example, agglomerative average link clustering (Jain and Dubes, 1988), k-means (Hartigan and Wong, 1979), highlights compactness. The resultant clusters using this algorithm is spherical clusters. Modification on k-means was discussed in (Arthur and Vassilvitskii, 2007) which is defined as k-means++. Though it uses a better centroid initialization technique for improvements over k-means, still it suffers from some drawbacks because we need to specify the number of clusters to be formed in advance and it assumes a fixed shape of clusters. After investigating different existing clustering algorithms, we can find that spectral clustering is more suitable for our task. It embeds sentences on a low-dimensional eigen space and performs clustering on the data points mapped to the low-dimensional embedding space. It does not assume any fixed shape of the cluster but rather emphasizes graph partitioning (Hamad and Biela, 2008) based

[1]https://fasttext.cc/docs/en/crawl-vectors.html

on connectedness among the vertices representing the data points. So, it is very useful when the shape of cluster is non-convex. Nowadays, the spectral clustering algorithm has been used in a wide range of application areas like image clustering (Tilton, 1998), shape clustering (Sidi et al., 2011), motion clustering (Lauer and Schnörr, 2009) and many more. Gupta et. al. (Gupta et al., 2019) presented a spectral clustering-based text summarization approach, which uses Textual Entailment(TE) and Spectral Clustering (ATESC) to calculate sentence connectedness scores. It is used to measure the saliency of a sentence in the input.
However, to the best of our knowledge, it is our new attempt to use a spectral clustering algorithm in the Bengali text summarization domain. For sentence representation, we have also considered sentence position as a new feature and combined it with the semantic content-based sentence features. The spectral clustering is applied to the sentence vectors to produce multiple clusters where each cluster represents a topical segment of the input document. The final summary is generated by choosing sentences from the ordered clusters using a centrality-based saliency measure (Günes and Dragomir R., 2004).

3 Our Proposed Methodology

Steps of our proposed system is illustrated in Figure 1. Each step of the proposed system is discussed in this section.

3.1 Preprocessing

Sentences are identified using a sentence tokenizer available with the NLTK toolkit. A sentence is split up into words. Stop words are discarded from the sentences. Stop words denotes unimportant frequent words in the dataset. A predefined, human-made list of Bengali words [2] was considered for stop word removal. 363 stop words were considered in that stop word list. A sample sentence after discarding stop words from it is represented in Figure 2.

3.2 Sentence Vectorization

After pre-processing, sentences are passed to the vectorization step. The vector for a sentence is obtained by taking an average of the vectors corresponding to the words that appeared in the sentence.

[2]http://fire.irsi.res.in/fire/static/resources

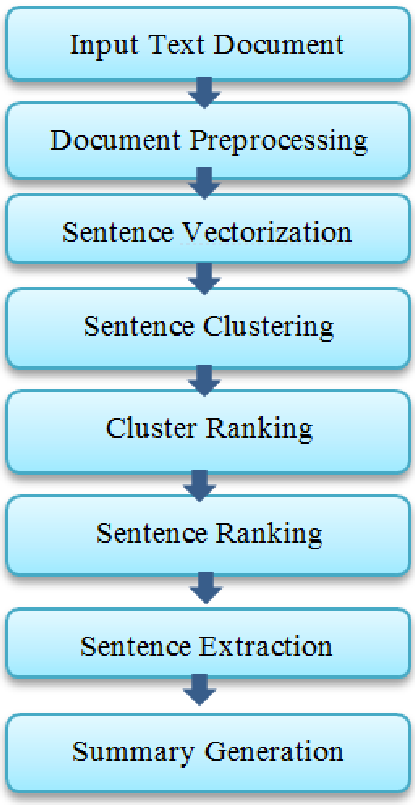


Figure 1: Steps of the proposed summarization system

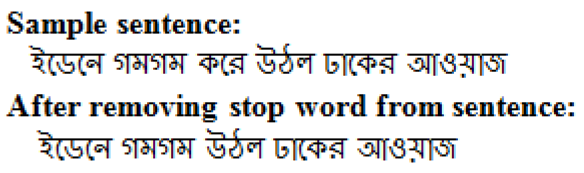

Sample sentence:
ইডেনে গমগম করে উঠল ঢাকের আওয়াজ
After removing stop word from sentence:
ইডেনে গমগম উঠল ঢাকের আওয়াজ

Figure 2: Removal of stop word for Bengali sentence

fastText word embeddings [3] were used to get word vectors. Since the size of a word vector is 300, the dimension of the sentence vector obtained using the average rule is 300. The value for the feature "sentence position" is appended at the end of the sentence vector, which increases its dimension to 301. The value for the sentence position feature is calculated as the division of the position of the sentence in the document by the total sentences in the document. Hence our final sentence vector is of dimension 301. The rationale behind including sentence position in the sentence vector is to encourage locally coherent sentences to fall in the same cluster. This helps to segment a document in a better way.

[3]`https://fasttext.cc/docs/en/crawl-vectors.html`

3.3 Sentence Clustering

In this step, sentence vectors are clustered into clusters of different sizes. The idea is to group similar and closer sentences into the same cluster. To cluster the sentences, we have used the spectral clustering algorithm. To implement spectral clustering, we first calculate the affinity matrix from a document graph in which a node corresponds to a sentence vector, and the edge between two nodes is weighted by the similarity between the corresponding two vectors. Affinity matrix is created using the similarity function given in Equation 1, which is basically a Gaussian similarity function.

$$A_{ij} = exp(\frac{-d^2(s_i, s_j)}{\sigma^2}) \tag{1}$$

Where σ is a control parameter that controls the context window in our case. In equation 1, $d(s_i, s_j)$ denotes distance between two sentence vectors s_i and s_j. Distance between two points (x_1, x_2) and (y_1, y_2) is calculated using the formula of Euclidean Distance defined in equation 2. In our approach, we varied $sigma$ and got the best result when it is set to 10.

$$dist = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{2}$$

From the affinity matrix, the graph Laplacian matrix is obtained using equation 3.

$$L = D - A \tag{3}$$

where A is the affinity matrix and D is the degree matrix such that

$$d_i = \sum_{j|(i,j)\in E} W_{ij} \tag{4}$$

where E is the set of edges in the graph and W_{ij} refers to the similarity between two points x_i and x_j corresponding to two different sentences in a document.
After normalizing the graph Laplacian matrix, the Eigen values and Eigen vectors of the normalized graph Laplacian matrix are used to embed the sentences into a low dimension Eigen space (Luxburg, 2007). Finally, a simple k-means clustering algorithm is applied for clustering the low dimensional dense vectors to obtain hard clusters. The main problem in the K-means cluster algorithm is that it needs to specify the value of K in advance. In our case, we have used an Eigen map heuristic method to determine the value of K. The main idea is to choose the value K such that all eigenvalues λ_1,.......λ_k are very small, but λ_{k+1} is relatively large. The details of this method can be found in (Luxburg, 2007). We have used this method to determine the number of clusters. Thus the number of topical segments is automatically inferred in an unsupervised way.

3.4 Cluster Ranking

The clusters are ranked in ascending order on the basis of the average of the position values of the sentences, present in that cluster. The cluster ranking enables us to identify the more significant clusters from which sentence extraction will occur first. The rationale behind using the position-based cluster ranking method is to ensure the selection of sentences in the summary from the topics in order as they appear in the text (position-based topical order). This is useful in creating an informative extract of sentences covering various topics in a document.

3.5 Within-cluster Sentence Ranking

A particular cluster may have multiple sentences present in it. To identify the most salient sentence from each cluster, the sentences within a particular cluster are ranked using the graph-based lexical centrality method published in (Günes and Dragomir R., 2004). In this method, a weighted adjacency matrix is constructed for the graph representing each cluster where the sentences in the cluster are considered as the vertices and the cosine similarity is considered as the edge weights between two sentence vectors. Cosine similarity is one of the popular similarity measures between two vectors. It is the cosine of the angle between two vectors, which means the dot product of two vectors divided by the product of their lengths. Cosine similarity is calculated using equation 5, where A and B are two sentence vectors belonging to a cluster. The rank of the sentence in a particular cluster is the sum of all the cosine edge weights to all other vertices in the cluster graph. The higher the sum of edge weights the higher the rank of the sentence is. This score is used to identify the sentence which is the most central to the cluster.

$$cosine - similarity = \frac{A.B}{||A||||B||} \tag{5}$$

For example, let us assume that six sentences are present in a cluster. Now adjacency matrix is created for that cluster using the cosine similarity

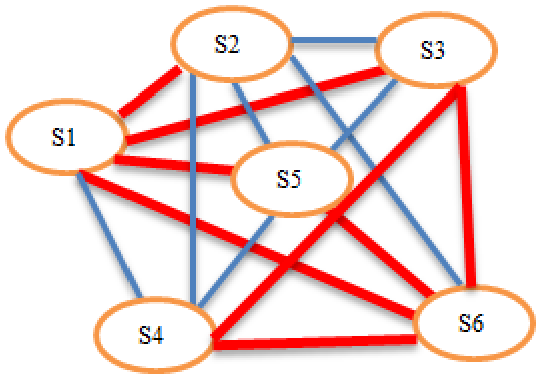


Figure 3: A sample similarity graph for the sentences in a cluster

value. A sample weighted matrix for the graph representing a cluster is as follows.

$$\begin{array}{c|cccccc} & s_1 & s_2 & s_3 & s_4 & s_5 & s_6 \\ \hline s_1 & 0 & 0.8 & 0.7 & 0.3 & 0.8 & 0.75 \\ s_2 & 0.8 & 0 & 0.3 & 0.2 & 0.4 & 0.3 \\ s_3 & 0.7 & 0.3 & 0 & 0.6 & 0.3 & 0.7 \\ s_4 & 0.3 & 0.2 & 0.6 & 0 & 0.2 & 0.8 \\ s_5 & 0.8 & 0.4 & 0.3 & 0.2 & 0 & 0.9 \\ s_6 & 0.75 & 0.3 & 0.7 & 0.8 & 0.9 & 0 \end{array}$$

In the above sentence similarity matrix, the row sum gives the sum of the similarities of a given sentence to the other sentences in the cluster they belong to. From the above sentence similarity matrix, we can observe that s_6 has the highest saliency score and it should be selected first as a summary worthy sentence from the cluster.

3.6 Summary Generation

After cluster ordering, sentences within each cluster are ranked. Then the sentence extraction process begins to create a summary for each input document. Here we select the first-ranked sentence in the first cluster followed by the first-ranked sentence in the second cluster and so on. If the number of clusters produced by the clustering algorithm is less than the required number of sentences, the process is repeated in a round-robin fashion until we obtain the required number of sentences. Once the required number of sentences is extracted to attain the desired summary length, the process is stopped. In our approach, a summary of 100 words is taken for evaluation. The algorithm for the overall summary generation process is shown in Algorithm 1.

Algorithm 1 Summary generation using spectral clustering based document segmentation

```
Input: A text document.
Output: Summary of the document.
 1: Breaking the input document into sentences.
 2: Removal of stop words from the sentences.
 3: Calculation of the 300-dimensional sentence vector by taking an average of word vectors obtained using the fastText open source.
 4: Calculate the positional feature value for each sentence and append it with the 300-dimensional sentence vector obtained in the previous step. The positional feature value is calculated as the position of the sentence in the document divided by the total number of sentences in the document.
 5: Sentence affinity matrix is created using the Gaussian similarity function.
 6: Compute the normalized graph Laplacian for the sentence affinity matrix.
 7: Eigenvalue decomposition is performed on the normalized graph Laplacian to get eigenvalues and eigenvectors. The eigenvectors are arranged in the ascending order of eigenvalues.
 8: Take the first d eigenvectors to form an N × d matrix U. A matrix T is obtained from U by normalizing the rows of U to norm 1.
 9: Eigen gap heuristic is applied to identify the gap which gives the optimal number of clusters, K.
10: Treating each row of T as a spectral embedding of a sentence, a simple K-means algorithm is applied on T to obtain K clusters where K is computed using the Eigen gap heuristic.
11: Clusters are ranked on the basis of the average position values of the sentences belonging to a cluster.
12: The sentences within each cluster are assigned scores based on the graph centrality-based saliency measure (Günes and Dragomir R., 2004).
13: For creating a summary, the top-ranked cluster contributes first its best sentence to the summary and then the second-ranked cluster contributes, and so on. If the number of clusters produced by the clustering algorithm is less than the required number of sentences, the process is repeated in a round-robin fashion until we obtain the required number of sentences.
```

4 Datasets and Experimental Results

Since no publicly available dataset is available for Bengali text summarization, we have tested our proposed approach on our own dataset consisting of 102 Bengali document-summary pairs. The average number of sentences in each document is 40. For evaluation, we have taken 100 words from each summary generated by the system using the proposed approach.

We have conducted six experiments to prove the effectiveness of our approach. These experiments include implementations of five state-of-the-art unsupervised methods with which our proposed approach is compared. A brief description of the models implemented by us is given below in this section.

Model 1: This is our proposed model that uses clustering-based document segmentation for text summarization.

Model 2: This approach was developed by Luhn (Luhn, 1958). It generates a summary from a document by considering that the sentence containing more frequent words is more important than the sentence containing less frequent words. In this method, stop words are removed before sentence weight calculation.

Model 3: This approach was developed by (Günes and Dragomir R., 2004). It uses Lexrank, which is a graph-based approach that represents sentences as vertices of a graph and considers the cosine similarity between any two sentences as the weight of the edge between the corresponding vertices. Finally, Google's page rank algorithm is applied to the graph to rank sentences.

Model 4: This approach was developed by (Ani and Lucy, 2005). The approach is named "Sumbasic" which considers term frequency as the saliency of a term. Here probability of a word is calculated based on its frequency and each sentence is assigned a score equal to the average probability of the words contained in the sentence. The sentence with the highest score is selected first in the summary. Before selecting the next sentence, the words present in the already selected sentence are penalized by multiplying their probability values by themselves, and the sentences are re-ranked using the newly calculated probability values. After re-ranking the sentences, the sentence with the highest score is selected as the second sentence of the summary. This process is continued until the summary of the desired length is obtained.

Model 5: This model was developed by (Rada and Paul, 2004). It is called as Textrank. It is also a graph-based approach similar to that used in LexRank(Günes and Dragomir R., 2004). LexRank used TF*IDF-based term weight and cosine similarity value as the edge weight whereas TextRank used word overlap-based similarity value as the edge weight.

Model 6: This is the lead baseline model, where a summary is generated by considering the first 100 words of the input document. This is the baseline defined in DUC 2001 and DUC 2002 shared tasks on single document summarization.

4.1 Summary Evaluation Metric

To calculate the performance score of the proposed model, we have used the popular summary evaluation package called ROUGE (Lin, 2004) which measures n-gram overlap between a system-generated summary and the reference summaries (Wan and Yang, 2006). In our case, we have used one reference summary for each system-generated summary. ROUGE counts various kinds of overlapping units between the system summary and the reference summaries. We have used the latest version of the ROUGE package - ROUGE 1.5.5 for evaluating the system summaries. The ROUGE toolkit reports various ROUGE–N scores, for example, ROUGE-1, ROUGE-2, etc. Along with ROUGE-1 scores, many state-of-the-art summarization systems have been evaluated using ROUGE-2 (bigram-based), and ROUGE-SU4 (skip bigrams with skip distance up to 4 words (Lin, 2004)). So, we consider ROUGE-1, ROUGE-2, and ROUGE-SU4 scores for evaluating our proposed summarization models. We set the summary length to 100 words by using the -l 100 option in the ROUGE toolkit, which takes the first 100 words from each system summary for evaluation. We use ROUGE-F score scores to evaluate and compare our proposed neural summarization method with other existing summarization methods.

4.2 Results and Comparisons

We have implemented five existing summarization systems for comparing them with the system proposed by us. The comparison results are shown in Table 1.

As we can see from the table 1, our proposed model (Model 1) performs significantly better than

MODEL	Rouge 1	Rouge 2	Rouge SU4
Model 1(Proposed Model)	**0.4481**	**0.2844**	**0.2848**
Model 2 (Luhn, 1958)	0.3929	0.2324	0.2293
Model 3 (Günes and Dragomir R., 2004)	0.3693	0.1995	0.1980
Model 4 (Ani and Lucy, 2005)	0.3602	0.1831	0.1836
Model 5 (Rada and Paul, 2004)	0.3499	0.1846	0.1835
Model 6(Lead Baseline)	0.2733	0.1501	0.1487

Table 1: Performance of our proposed summarization model and its comparison with some existing summarization methods

other baseline models to which it is compared. The proposed model also performs significantly better than Model 3 (Günes and Dragomir R., 2004) which uses the graph-based ranking of all sentences of the input document considering all sentences in the document as a single cluster. Compared to the system "LexRank " (Günes and Dragomir R., 2004), we use spectral clustering-based document segmentation and within-cluster sentence ranking. It is evident from the results that, instead of taking the input document as a single cluster of sentences, if the document is segmented into multiple topical clusters and the summary is generated by choosing sentences from the clusters one by one, this produces a summary which is better in quality than that produced by the system called "LexRank".

We have computed performance improvement using equation 6 which computes the difference between ROUGE scores obtained by the proposed model and the model to which it is compared.

$$PI = M - N \tag{6}$$

where PI denotes performance improvement, M is the score for the proposed approach, and N is the score for the approach to which the proposed approach is compared. Performance Improvement of our proposed approach over other approaches is shown in Table 2.

MODEL	Rouge 1	Rouge 2
Model 2	0.0552	0.052
Model 3	0.0788	0.0849
Model 4	0.0879	0.1013
Model 5	0.0982	0.0998
Model 6	0.1748	0.1343

Table 2: Performance improvement achieved by our proposed model in comparison with some state-of-the art summarization methods

As we can see from Table 2, the proposed approach shows improvement over the lead baseline and the LexRank(Günes and Dragomir R., 2004) by 0.1748 and 0.0788 ROUGE-1 points respectively.

4.2.1 Comparison of the spectral clustering algorithm with another conventional clustering algorithm

To prove the effectiveness of the spectral clustering algorithm in producing topical segments, we have implemented a variant of the proposed by replacing the spectral clustering algorithm with another popular clustering algorithm called DBSCAN. which is a density-based clustering algorithm(DBSCAN). DBSCAN algorithm is known to be robust to outliers. Minpts(minimum number of points for a cluster) and epsilion are two parameters that are tuned to achieve better performance. The best results are achieved by setting minpts=5 and epsilion=0.4. The summarization evaluation scores are shown in Table 3. It is evident from these results that spectral clustering is more effective for segmenting a document into multiple topics.

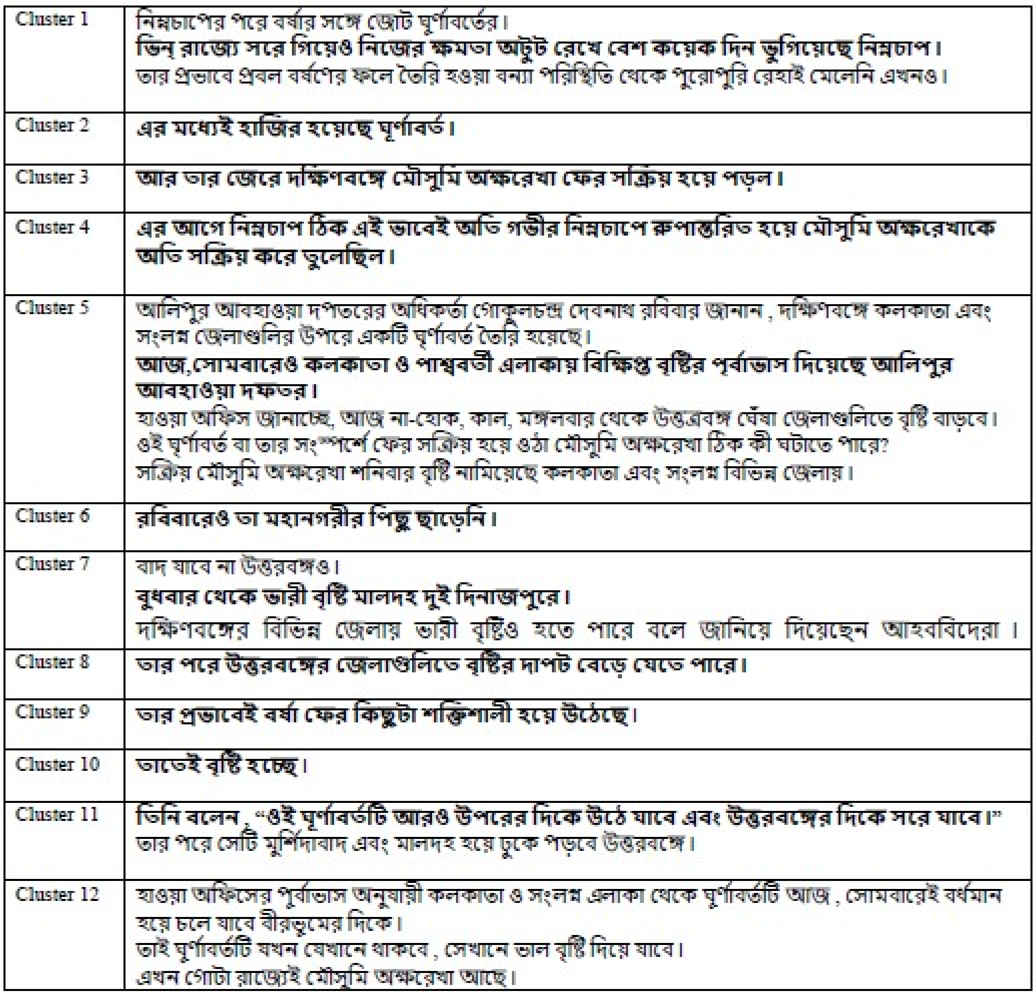
MODEL	Rouge 1	Rouge 2
Proposed Model	**0.4481**	**0.2844**
DBSCAN	0.3765	0.2253

Table 3: Comparison of the proposed model with spectral clustering with its variant that uses the DBSCAN clustering algorithm

4.3 An Example

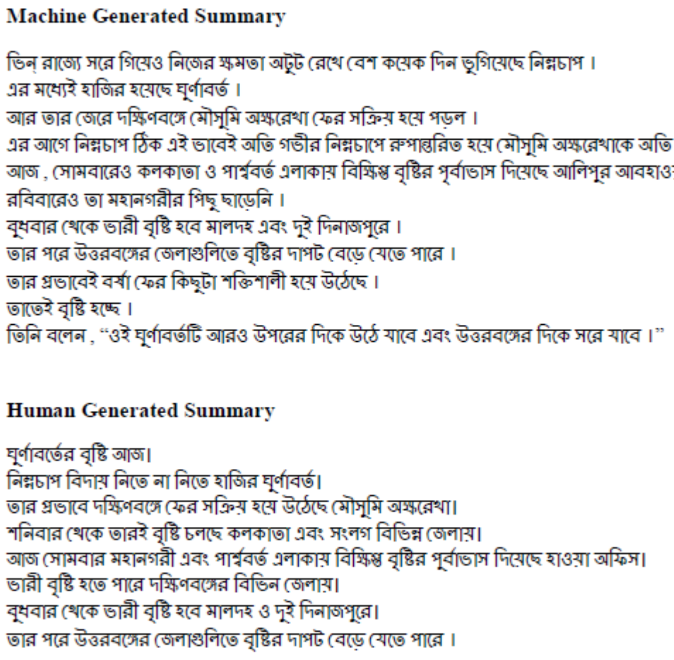
In this subsection, We have shown the clustering and the summarization results for an example input Bengali input document. The clusters produced by the spectral clustering algorithm, a reference summary and the system generated summary are shown in Figure 4 and 5 respectively.

The bold sentences in Figure 4 are the sentences selected by the summarization model proposed by us. Though we have shown in Figure 5 a system



Cluster 1	নিম্নচাপের পরে বর্ষার সঙ্গে জোট ঘূর্ণাবর্তের। **ভিন্ রাজ্যে সরে গিয়েও নিজের ক্ষমতা অটুট রেখে বেশ কয়েক দিন ভুগিয়েছে নিম্নচাপ।** তার প্রভাবে প্রবল বর্ষণের ফলে তৈরি হওয়া বন্যা পরিস্থিতি থেকে পুরোপুরি রেহাই মেলেনি এখনও।
Cluster 2	**এর মধ্যেই হাজির হয়েছে ঘূর্ণাবর্ত।**
Cluster 3	**আর তার জেরে দক্ষিণবঙ্গে মৌসুমি অক্ষরেখা ফের সক্রিয় হয়ে পড়ল।**
Cluster 4	**এর আগে নিম্নচাপ ঠিক এই ভাবেই অতি গভীর নিম্নচাপে রুপান্তরিত হয়ে মৌসুমি অক্ষরেখাকে অতি সক্রিয় করে তুলেছিল।**
Cluster 5	আলিপুর আবহাওয়া দপ্তরের অধিকর্তা গোকুলচন্দ্র দেবনাথ রবিবার জানান , দক্ষিণবঙ্গে কলকাতা এবং সংলগ্ন জেলাগুলির উপরে একটি ঘূর্ণাবর্ত তৈরি হয়েছে। **আজ,সোমবারেও কলকাতা ও পার্শ্ববর্তী এলাকায় বিক্ষিপ্ত বৃষ্টির পূর্বাভাস দিয়েছে আলিপুর আবহাওয়া দফতর।** হাওয়া অফিস জানাচ্ছে, আজ না-হোক, কাল, মঙ্গলবার থেকে উত্তরবঙ্গ ঘেঁষা জেলাগুলিতে বৃষ্টি বাড়বে। ওই ঘূর্ণাবর্ত বা তার সংস্পর্শে ফের সক্রিয় হয়ে ওঠা মৌসুমি অক্ষরেখা ঠিক কী ঘটাতে পারে? সক্রিয় মৌসুমি অক্ষরেখা শনিবার বৃষ্টি নামিয়েছে কলকাতা এবং সংলগ্ন বিভিন্ন জেলায়।
Cluster 6	**রবিবারেও তা মহানগরীর পিছু ছাড়েনি।**
Cluster 7	বাদ যাবে না উত্তরবঙ্গও। **বুধবার থেকে ভারী বৃষ্টি মালদহ দুই দিনাজপুরে।** দক্ষিণবঙ্গের বিভিন্ন জেলায় ভারী বৃষ্টিও হতে পারে বলে জানিয়ে দিয়েছেন আবহবিদেরা ।
Cluster 8	**তার পরে উত্তরবঙ্গের জেলাগুলিতে বৃষ্টির দাপট বেড়ে যেতে পারে।**
Cluster 9	**তার প্রভাবেই বর্ষা ফের কিছুটা শক্তিশালী হয়ে উঠেছে।**
Cluster 10	**তাতেই বৃষ্টি হচ্ছে।**
Cluster 11	**তিনি বলেন , "ওই ঘূর্ণাবর্তটি আরও উপরের দিকে উঠে যাবে এবং উত্তরবঙ্গের দিকে সরে যাবে।"** তার পরে সেটি মুর্শিদাবাদ এবং মালদহ হয়ে ঢুকে পড়বে উত্তরবঙ্গে।
Cluster 12	হাওয়া অফিসের পূর্বাভাস অনুযায়ী কলকাতা ও সংলগ্ন এলাকা থেকে ঘূর্ণাবর্তটি আজ , সোমবারেই বর্ধমান হয়ে চলে যাবে বীরভূমের দিকে। তাই ঘূর্ণাবর্তটি যখন যেখানে থাকবে , সেখানে ভাল বৃষ্টি দিয়ে যাবে। এখন গোটা রাজ্যেই মৌসুমি অক্ষরেখা আছে।

Figure 4: Clusters produced using spectral clustering algorithm



Machine Generated Summary

ভিন্ রাজ্যে সরে গিয়েও নিজের ক্ষমতা অটুট রেখে বেশ কয়েক দিন ভুগিয়েছে নিম্নচাপ ।
এর মধ্যেই হাজির হয়েছে ঘূর্ণাবর্ত ।
আর তার জেরে দক্ষিণবঙ্গে মৌসুমি অক্ষরেখা ফের সক্রিয় হয়ে পড়ল ।
এর আগে নিম্নচাপ ঠিক এই ভাবেই অতি গভীর নিম্নচাপে রুপান্তরিত হয়ে মৌসুমি অক্ষরেখাকে অতি সক্রিয় করে তুলেছিল ।
আজ , সোমবারেও কলকাতা ও পার্শ্ববর্ত এলাকায় বিক্ষিপ্ত বৃষ্টির পূর্বাভাস দিয়েছে আলিপুর আবহাওয়া দফতর ।
রবিবারেও তা মহানগরীর পিছু ছাড়েনি ।
বুধবার থেকে ভারী বৃষ্টি হবে মালদহ এবং দুই দিনাজপুরে ।
তার পরে উত্তরবঙ্গের জেলাগুলিতে বৃষ্টির দাপট বেড়ে যেতে পারে ।
তার প্রভাবেই বর্ষা ফের কিছুটা শক্তিশালী হয়ে উঠেছে ।
তাতেই বৃষ্টি হচ্ছে ।
তিনি বলেন , "ওই ঘূর্ণাবর্তটি আরও উপরের দিকে উঠে যাবে এবং উত্তরবঙ্গের দিকে সরে যাবে ।"

Human Generated Summary

ঘূর্ণাবর্তের বৃষ্টি আজ।
নিম্নচাপ বিদায় নিতে না নিতে হাজির ঘূর্ণাবর্ত।
তার প্রভাবে দক্ষিণবঙ্গে ফের সক্রিয় হয়ে উঠেছে মৌসুমি অক্ষরেখা।
শনিবার থেকে তারই বৃষ্টি চলছে কলকাতা এবং সংলগ বিভিন্ন জেলায়।
আজ সোমবার মহানগরী এবং পার্শ্ববর্ত এলাকায় বিক্ষিপ্ত বৃষ্টির পূর্বাভাস দিয়েছে হাওয়া অফিস।
ভারী বৃষ্টি হতে পারে দক্ষিণবঙ্গের বিভিন জেলায়।
বুধবার থেকে ভারী বৃষ্টি হবে মালদহ ও দুই দিনাজপুরে।
তার পরে উত্তরবঙ্গের জেলাগুলিতে বৃষ্টির দাপট বেড়ে যেতে পারে ।

Figure 5: System generated summary and reference summary of the document

generated summary consisting of 11 sentences, the first 100 words of it is taken during evaluation using the ROUGE package.

5 Conclusion

This paper describes a spectral clustering-based method for segmenting a document into multiple topical segments and a summarization method that generates an extractive summary by choosing sentences from the clusters. Our proposed summarization approach outperforms several existing baseline summarization approaches.
The higher ROUGE score obtained by the proposed approach proves that the spectral clustering algorithm provides more accurate topical segments of a document if the sentence position is added as an additional dimension to the sentence vector obtained by averaging the word vectors.
We have a future plan to use more deep semantic methods for document segmentation and incorporate them into the text summarization process.

References

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.

Nenkova Ani and Vanderwende Lucy. 2005. The impact of frequency on summarization. *Computer Science*.

David Arthur and Sergei Vassilvitskii. 2007. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA. Society for Industrial and Applied Mathematics.

Phyllis B. Baxendale. 1958. Machine-made index for technical literature - an experiment. *IBM J. Res. Dev.*, 2:354–361.

H. P. Edmundson. 1969. New methods in automatic extracting. *J. ACM*, 16(2):264–285.

Erkan Günes and Radev Dragomir R. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479.

Anand Gupta, Manpreet Kaur, Ahsaas Bajaj, and Ansh Khanna. 2019. Entailment and spectral clustering based single and multiple document summarization. *International Journal of Intelligent Systems and Applications*, 11:39–51.

Denis Hamad and Philippe Biela. 2008. Introduction to spectral clustering. In *Introduction to spectral clustering*, pages 1 – 6.

J. A. Hartigan and M. A. Wong. 1979. Algorithm AS 136: A K-Means clustering algorithm. *Applied Statistics*, 28(1):100–108.

Anil K. Jain and Richard C. Dubes. 1988. *Algorithms for Clustering Data*. Prentice-Hall, Inc., USA.

Yaochu Jin. 2006. *Multi-objective machine learning*, volume 16. Springer Science & Business Media.

Hongyan Jing and Kathleen R. McKeown. 2000. Cut and paste based text summarization. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, NAACL 2000, page 178–185, USA. Association for Computational Linguistics.

Fabien Lauer and Christoph Schnörr. 2009. Spectral clustering of linear subspaces for motion segmentation. In *2009 IEEE 12th International Conference on Computer Vision*, pages 678–685.

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.

H. P. Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165.

Ulrike Luxburg. 2007. A tutorial on spectral clustering. *Statistics and Computing*, 17:395–416.

Mihalcea Rada and Tarau Paul. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Kamal Sarkar. 2008. Syntactic sentence compression: Facilitating web browsing on mobile devices. In *2008 International Conference on Information Technology*, pages 283–286.

Kamal Sarkar. 2009a. Centroid-based summarization of multiple documents. *TECHNIA–International Journal of Computing Science and Communication Technologies*, 2.

Kamal Sarkar. 2009b. Sentence clustering-based summarization of multiple text documents. *TECHNIA – International Journal of Computing Science and Co mmunication Technologies*, 2:325–335.

Kamal Sarkar. 2010. Syntactic trimming of extracted sentences for improving extractive multi-document summarization. *Journal of Computing*, 2:177–184.

Kamal Sarkar. 2012a. An approach to summarizing bengali news documents. In *International Conference on Advances in Computing, Communications and Informatics*, ICACCI '12, page 857–862, New York, NY, USA. Association for Computing Machinery.

Kamal Sarkar. 2012b. Bengali text summarization by sentence extraction. *CoRR*.

Kamal Sarkar and Sivaji Bandyopadhyay. 2005. Generating headline summary from a document set. In *Proceedings of the 6th International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing'05, page 649–652, Berlin, Heidelberg. Springer-Verlag.

Oana Sidi, Oliver van Kaick, Yanir Kleiman, Hao Zhang, and Daniel Cohen-Or. 2011. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. *ACM Trans. Graph.*, 30(6):1–10.

J.C. Tilton. 1998. Image segmentation by region growing and spectral clustering with a natural convergence criterion. In *IGARSS '98. Sensing and Managing the Environment. 1998 IEEE International Geoscience and Remote Sensing. Symposium Proceedings. (Cat. No.98CH36174)*, volume 4, pages 1766–1768 vol.4.

X. Wan and J. Yang. 2008. Multi-document summarization using cluster-based link analysis. In *SIGIR-08*, page 299–306.

Xiaojun Wan and Jianwu Yang. 2006. Improved affinity graph based multi-document summarization. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 181–184, New York City, USA. Association for Computational Linguistics.

Author Index