A Code-Mixed Dialog System in Medical Domain

Thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science and Engineering

by

Suman Dowlagar 20162152

suman.dowlagar@research.iiit.ac.in



International Institute of Information Technology Hyderabad - 500 032, INDIA July 2023

Copyright © Suman Dowlagar, 2023 All Rights Reserved

International Institute of Information Technology Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "A Code-Mixed Dialog System in Medical Domain" by Suman Dowlagar, Roll no: 20162152, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date

Adviser: Prof. Radhika Mamidi

To My Loving Husband

Acknowledgments

I want to express my deep gratitude to my supervisor Dr. Radhika Mamidi, Associate professor, IIIT-Hyderabad, for allowing me to do my Ph.D. research and providing continuous support, guidance, and encouragement throughout the research. I must be very fortunate to have an advisor who gave me the freedom to explore independently. She keeps inspiring me to push myself to achieve huge impacts on society by researching in the natural language processing field. At the same time, her guidance helped me recover whenever my steps faltered. Her vast knowledge in Computational Linguistics, especially in Dialog Systems helped me build strong papers, which contributed to completing the requirements for my PhD.

I am indebted to Language Technologies and Research Center (LTRC) faculty - Prof Dipti Misra Sharma, Dr. Soma Paul, and Dr. Manish Shrivastava for guiding me through the basic concepts of natural language processing and computational linguistics that helped me to understand my research area better. They also helped me understand the basic tools that played a crucial role in my research work.

I thank the doctors from Mahatma Gandhi Memorial Hospital (MGMH), Warangal, Telangana, India, for their help in data acquisition. I personally thank Dr. D. Satyanarayana, Retd. HOD, MGMH, Warangal, Telangana, Dr. N. Surender Reddy, MGMH, Warangal, Telangana, Dr. D. Aswini, Dr. PVK Shastry, Retd. HOD, MGMH, Warangal, and Dr. Ramchander Dharak, Retd. HOD, MGMH, Warangal for their assistance in the data collection, deciding slot annotation guidelines for medical terms, and their help in annotating the data.

Additionally, this research would not have been possible without the generous support from the "Touchstone" KCIS Enabler Grant Project, who financed my research. I thank the team.

I am also grateful to my friends and annotators who helped me to create the corpus necessary for my research, and also their guidance helped me to understand various concepts which benefited my thesis. Most significantly, none of this would have been possible without the love and patience of my family. I formally thank them for their support and strength through these years. I would like to thank my husband personally. From the beginning, he was the one who had confidence in me and guided me through the difficult times in my life, and never let me fail. I thank him immensely for that.

Abstract

In the healthcare domain, medical and patient interactions are crucial for diagnosis. Initially, AI models developed for healthcare centered only on monolingual data. However, such models do not cater to the multilingual regions, where most conversations are Code-Mixed. Our thesis concentrates on developing the Code-Mixed Medical Task-Oriented Dialog System. This task-oriented dialog system aims to help the user consult a medical specialist based on their symptoms. We train the dialog system on the code-mixed task-oriented dialog dataset 'Su-Vaid'. The dataset contains 3005 Telugu-English Code-Mixed dialogues between patients and doctors with 29k utterances covering ten specializations with an average code-mixing index (CMI) of 33.3%.

The major components of our dialog system are Natural Language Understanding (NLU), Dialog Manager (DM), and Natural Language Generation (NLG) modules. We manually annotated the conversational dataset with intent and slot labels. We also present baselines to establish benchmarks on the dataset using existing state-of-the-art Natural Language Understanding (NLU) models. We improved the existing baselines using contextual ground truth intent labels and processing the slots as chunks. We have used Rule-based Dialog Manager and template-based NLG module in our Dialog System.

While interacting with the system, the user may fail to mention a few symptoms. In such a case, the dialog system must remind the user by suggesting relevant symptoms. We have incorporated a suggestion system into our dialog system. Apart from interacting with the user to recommend the medical specialist, the dialog system needs to be more human-like (empathetic). Also, human-human conversations in the healthcare domain are mostly empathetic. To achieve human-like behavior, we have included empathy via language accommodation in our dialog system.

Also, interactions between patients and medical practitioners can sometimes be challenging. Health care workers are at greater risk of workplace abuse than most other workers, with nurses and family physicians rated as most at risk of abusive encounters with patients. Types of abusive encounters range in severity, from verbal threats to more extreme encounters, such as stalking and physical assault. Hate speech and physical abuse must be strongly condemned for a better function of health care. Such conversations must be identified by the system so that the system can take necessary action on such users. So we have included hate speech detection in our dialog system.

Finally, we have evaluated all the major components of our dialog system using state-of-the-art baselines and classification metrics and we chose the top performing models to integrated them into the final version of the system.

Related Publications

Journals

- 1. Dowlagar, Suman, and Radhika Mamidi. "A code-mixed task-oriented dialog dataset for the medical domain." *Computer Speech & Language* 78 (2023): 101449.
- 2. Dowlagar, Suman, and Radhika Mamidi. "Hate Speech Detection on Code-Mixed Dataset Using a Fusion of Custom and Pre-trained Models with Profanity Vector Augmentation." *SN Computer Science* 3, no. 4 (2022): 1-17.

Conference Papers Related to Thesis

- 1. Dowlagar Suman and Radhika Mamidi. "A Recommendation System in Task-Oriented Doctor-Patient Interactions." *Women In Machine Learning Workshop, NeurIPS*. 2022
- Dowlagar Suman and Radhika Mamidi. "A Semantically Conditioned Code-Mixed Natural Language Generation for Task-Oriented Dialog." Women In Machine Learning Workshop, NeurIPS. 2022
- Dowlagar Suman and Radhika Mamidi. "A Survey of Recent Neural Network Models on Code-Mixed Indian Hate Speech Data" *The* 13th meeting of Forum for Information Retrieval Evaluation (FIRE) 2021.
- 4. Dowlagar Suman, and Radhika Mamidi. "DepressionOne@ LT-EDI-ACL2022: Using Machine Learning with SMOTE and Random UnderSampling to Detect Signs of Depression on Social Media Text." In Proceedings of the Second Workshop on Language Technology for Equality, Diversity, and Inclusion, Association for Computational Linguistics, pp. 301-305. 2022.
- Dowlagar Suman, and Radhika Mamidi. "OFFLangOne@ DravidianLangTech-EACL2021: Transformers with the Class Balanced Loss for Offensive Language Identification in Dravidian Code-Mixed text." In Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages, European Chapter of the Association for Computational Linguistics, pp. 154-159. 2021.

- Dowlagar Suman, and Radhika Mamidi. "HASOCOne@ FIRE-HASOC2020: Using BERT and Multilingual BERT models for Hate Speech Detection." in The 12th meeting of Forum for Information Retrieval Evaluation (FIRE) 2020.
- Dowlagar Suman and Radhika Mamidi. Computational Linguistics and Intelligent Text Processing: 17th International Conference, CICLing 2016, Konya, Turkey, April 3-9, 2016, Proceedings, Part I, chapter "A Karaka Dependency based Dialog Act Tagging for Telugu using Combination of LMs and HMM." Springer International Publishing, Cham, 2016.
- Dowlagar Suman and Radhika Mamidi. "A semi-supervised dialog act tagging for Telugu." In ICON 2015: 12th International Conference on Natural Language Processing, 2015

Conference Papers Related to Code-Mixing

- Dowlagar Suman, Radhika Mamidi. "A Pre-trained Transformer and CNN model with Joint Language ID and Part-of-Speech Tagging for Code-Mixed Social-Media Text." Recent Advances in Natural Language Processing (RANLP), 2021.
- 2. Dowlagar Suman, Radhika Mamidi. "Gated Convolutional Sequence to Sequence Based Learning for English-Hingilsh Code-Switched Machine Translation." Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching, NAACL 2021.
- 3. Dowlagar Suman and Radhika Mamidi. "*Graph convolutional networks with multi-headed attention for code-mixed sentiment analysis.*" Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages, EACL 2021.
- 4. Dowlagar Suman, Radhika Mamidi. "EDIOne@ LT-EDI-EACL2021: Pre-trained Transformers with Convolutional Neural Networks for Hope Speech Detection." Proceedings of the First Workshop on Language Technology for Equality, Diversity, and Inclusion, EACL 2021.
- 5. Dowlagar Suman and Radhika Mamidi. "CMSAOne@Dravidian-CodeMix-FIRE2020: A Meta Embedding and Transformer model for Code-Mixed Sentiment Analysis on Social Media Text." FIRE 2020.

Other Papers

- 1. Dowlagar Suman and Radhika Mamidi. "Does a Hybrid Neural Network based Feature Selection Model Improve Text Classification?" ICON 2020
- 2. Dowlagar Suman and Radhika Mamidi. "Multilingual Pre-Trained Transformers and Convolutional NN Classification Models for Technical Domain Identification." ICON 2020

3. Dowlagar Suman and Radhika Mamidi. "Unsupervised Technical Domain Terms Extraction using Term Extractor." ICON 2020

Contents

Ch	apter		Page
1	Intro	duction	. 1
	1.1	Code-Mixing Phenomenon	1
	1.2	Task-Oriented Dialog System	6
		1.2.1 Natural Language Understanding	7
		1.2.2 Hate Speech Detection	7
		1.2.3 Dialog Manager	9
		1.2.4 Suggestion System	9
		1.2.5 Medical Specialist Classification	10
		1.2.6 Natural Language Generation	10
	1.3	Motivation	10
	1.4	Challenges of Data collection and Annotation	11
	1.5	Major Contributions	12
	1.6	Organization of the thesis	13
2	Liter	rature Survey	. 15
_	2.1	Dialog Datasets	15
	2.2	Code-Mixing in Natural Language Processing	15
	2.3	Code-Mixed Hate Speech Detection	18
	2.4	Natural Language Understanding	18
	2.5	Dialog Management	19
	2.6	Recommender System	20
	2.7	Medical Specialist Classification	22
	2.8	Natural Language Generation	22
	2.9	Summary	23
3	A Co	ode-Mixed Task-Oriented Dialog Dataset for Medical Domain	. 24
-	3.1	Data Collection	24
	3.2	Data Conversion	27
	3.3	Code-Mixing Index	28
	3.4	Data Analysis	28
		3.4.1 Medical Conceptual Analysis	29
		3.4.2 Conversational Analysis	30
	3.5	Role of code-mixing in our dataset	33
	3.6	Annotation	36
		3.6.1 Intents	36

CONTENTS

		3.6.2	Slots	37
		3.6.3	Annotation Analysis	39
		3.6.4	Should we do Over-sampling and Under-sampling to handle the imbalanced data?	39
	3.7	Summa	ury	41
4	Natu	iral Lang	uage Understanding	42
	4.1	NLU m	10dels	42
		4.1.1	Independent Models	42
		4.1.2	Joint Models	43
		4.1.3	Contextual models	44
	4.2	Experir	nental Settings	45
	4.3	Perform	nance of Multilingual Models	45
	4.4	Cross-l	ingual XLMR Models	47
	4.5	Perform	nance of Cross-lingual NN models	48
	4.6	Context	tual Signals and Chunk Identification for NLU	48
		4.6.1	mBERT with Contextual Signal for IC	48
		4.6.2	Chunk Identification - SF Model	49
		4.6.3	Evaluation of our approach	51
		4.6.4	Adapter-Transformers	54
	4.7	Summa	ury	57
5	Hate	Speech	Detection	58
	5.1	Propose	ed Models for Code-Mixed Hate Speech Detection	59
		5.1.1	Transformers with the Class Balanced Loss for Offensive Language Identifica-	
			tion in Dravidian Code-Mixed text.	59
		5.1.2	Using BERT and multilingual BERT models for hate speech detection (with	
			data augmentation)	60
		5.1.3	Hate Speech Detection on Code-Mixed Dataset Using a Fusion of Custom and	
			Pre-trained Models with Profanity Vector Augmentation	61
	5.2	Dataset	is	65
	5.3	Experir	nental settings	67
	5.4	Baselin	les	67
		5.4.1	Support Vector Machines	68
		5.4.2	FastText	68
		5.4.3	CNN	68
		5.4.4	Bi-LSTM	68
		5.4.5	Multilingual BERT (mBERT)	68
		5.4.6	XLM-RoBERTa (XLMR)	68
		5.4.7	Genetic Algorithm	69
		5.4.8	AWD-LSTM with mBERT + XLMR	69
	5.5	Perforn	nance	69
		5.5.1	Ablation study	71
		5.5.2	Confusion matrix	73
		5.5.3	Training Loss	75
	5.6	HSD in	Dialog System	77
	5.7	Summa	ary	77
			· · · · · · · · · · · · · · · · · · ·	-

CONTENTS

6	Dial	og Manager and Suggestion System	8
	6.1	Dialog Manager	8
	6.2	Suggestion System	0
	6.3	Content Based Collaborative Filtering Models	4
		6.3.1 TF-IDF based collaborative Filtering	4
		6.3.2 SVD-based Collaborative Filtering	4
		6.3.3 Collaborative filtering using Deep Learning Methods	1
	61	Evaluation of Suggestion system	т Л
	6.5		+ 7
	0.5	Summary	/
7	Med	ical Specialist Classification	8
	7.1	Classification Models	8
		7.1.1 Extra Trees Classifier(ETC)	9
		712 CNN 89	9
		713 Bi-LSTM 89	9
		714 factText 0	ó
		7.1.5 Taxt CCN 00	0
		7.1.5 TEXT OCN	0
		7.1.0 IIIDENI	0
	7 0	/.I./ ALM-K	1
	1.2	Experimental settings	1
	7.3		2
	7.4	Summary	3
8	Natu	ral Language Generation	4
	8.1	Input to NLG	4
	8.2	NLG Models 94	5
	0.2	8 2 1 Template-Based NI G	5
		8.2.2 Machine Learning-Based NLG	5
	83	Baseline Models	5 6
	0.5	8.2.1 TempleteNLC	0 6
		8.3.1 Templatenico	0 7
		8.3.2 LSTM for language Generation	/
		8.3.3 Convolutional Seq2seq model	/
		8.3.4 Transformer model	/
		8.3.5 XLM	7
		8.3.6 mBART	7
		8.3.7 T5	8
		8.3.8 GPT 2	8
	8.4	Experimental setup	8
	8.5	Summary	9
0	The	Code Mixed Task Oriented Dielog System	Λ
7	0.1	$ \begin{array}{c} \text{Climpso about the system} \\ 100 \end{array} $	0
	9.1	Comparing our system with ChetCDT	0
	9.2	Comparing our system with ChatGP1	9
10	Cone	clusion and Future Work	4
	10.1	Summary	4
	10.2	Future Work	4

CONTENTS

Apper	ndix A: Appendix	х		•				•					•			•							11	6
A.1	Intent and Slot 7	Fables		•	 •	 •	•		 •	•	• •		•			•	•	•••	•		•	•	11	7
Bibliogra	phy			•	 •			•	 •				•		•	•	•			•			12	0

List of Figures

Figure		Page
1.1	Modular Task-Oriented Dialog System Architecture	6
3.1	Stages of data collection, data annotation, data analysis, data modeling, and final pre-	
	dictions	24
3.2	The distribution of dialogs with respect to medical specialists in the dataset	25
3.3	Intents and their distribution in the corpus	36
3.4	Top 15 slots and their distribution in the corpus	37
4.1	mBERT with contextual signal for Intent Classification	49
4.2	Bourndary aware Slot Filling Model using 2 phase mechanism. tp, s, and pos refer to	
	time_period, statement, and positive tags, respectively	50
4.3	Comparision between BERTNLU-context and mBERT with the contextual signal on the	
		51
4.4	Confusion matrix - BERTNLU-context for intent tagging.	52
4.5	Confusion matrix - mBERT with contextual signal for intent tagging.	53
4.6	Performance of naive mBERT and chunk identification SF models on the dataset for	51
4 7	each slot label (Only a lew slots were used for clear observation)	54
4.7	Overall accuracy performance per epoch for naive mBERT and chunk identification SF	55
		33
5.1	patient's using hate speech on doctors	59
5.2	Training the code-mixed model from scratch on the code-mixed dataset	62
5.3	The proposed method	64
5.4	Comparison of mBERT baseline and Our Approach on Hindi-English datasets	73
5.5	Comparison of mBERT baseline and Our approach on Dravidian datasets	74
5.6	Comparison of Training loss of our approach (with or without profanity vector) and the	
	mBERT model on the hate speech datasets	76
6.1	Patient's frame to be tracked and updated by the dialog system via user interaction	79
6.2	Doctor suggesting symptoms to a patient during an interaction.	81
6.3	Suggestion System Model.	81
7.1	F1-score for each specialist label when mBERT classifier is used	91
7.2	Confusion matrix for specialist classification task - mBERT	92

LIST OF FIGURES

8.1	Delexicalization for NLG data generation.	96
9.1	Delexicalization for NLG data generation.	100
9.2	An interaction with the dialog system with focus on code-mixing	102
9.3	An interaction with the dialog system with focus on the English language	103
9.4	An interaction with the dialog system with focus on the Telugu language	104
9.5	An interaction with the dialog system for hate speech detection	105
9.6	An interaction with the dialog system where all information is given at once	106

List of Tables

Page

Table

1.1	A Code-Mixed Telugu-English doctor-patient conversation. S, d, p refers to speaker, doctor, and patient, respectively. The Telugu script is written in wx format and italicized. Translations are given in brackets	2
1.2	A CM Telugu-English Utterance, with EN being English and TE being Telugu languages.	$\frac{2}{3}$
3.1	Statistics of our dataset	25
3.2	The proportion of languages in the corpus at dialog, utterance, and word/token levels	26
3.3	A doctor-patient conversation	29
3.4	A doctor-patient conversation	30
3.5	Sub dialogs in the dataset. S refers to the speaker of the data, d refers to the doctor's	
	utterance and p refers to the patient's utterance.	32
3.6	Examples of code-mixed utterances taken from the corpus with their monolingual coun-	24
0.7	terparts.	34
3.7	Intent and slot annotation in our dataset	38
4.1	Performance of Baseline models on our dataset, Where I refers to Independent Model, J refers to Joint Model, M refers to Memory Model, IC refers to Intent Classification, SE refers to slot Filling and P. P. Fl. A stands for macro Precision macro Precision	
4.2	macro-F1 measure, and accuracy	46
4.2	pendent Models, J stands for Joint Models, and M stands for Memory Models. P, R, F1,	17
12	A stands for macro-Precision, macro-Recall, and macro-F1 measure, and accuracy	4/
4.3 4.4	performance of our IC and SF models on the dataset	51
	score and Accuracy score, respectively.	56
5.1	Dataset Statistics	65
5.2	pared to our approaches on the baselines without and with profanity vector (PV) when com-	70
5.3	The ablation study of our model by using "accuracy score" as the investigation metric on the code-mixed hate speech datasets.	72
6.1	Symptoms selected from four interactions in the clinical data	82

LIST OF TABLES

6.2	Performance of the content-based recommendation systems with collaborative filtering	
	on our dataset	85
6.3	Variations of single symptom	86
7.1	Performance of text classification models on the code-mixed medical dataset	91
8.1	Performance of NLG module	99
9.1	A detailed working of the dialog system	109
9.2	Comparing the performance of chatGPT based on our input given in Figure 9.2	110
9.3	Comparing the performance of chatGPT based on our input given in Figure 9.3	111
9.4	Comparing the performance of chatGPT based on our input given in Figure 9.6	112
9.5	Differences between our system and ChatGPT	113
A.1	Intent and their percentage in our corpus	117
A.2	Slots with their definitions, examples and its percentage in the corpus	119

Chapter 1

Introduction

Giving machines the ability to communicate with humans has been one of the fundamental objectives in the field of natural language processing. As compelling as it is challenging, developing dialogue systems capable of satisfying the end-user on a par with human-human interaction remains an elusive target. Narrower in scope than general-purpose conversational assistants, task-oriented dialogue (ToD)¹ systems have attracted interest in research and business communities.

New milestones are reached across natural language applications with the growing viability of deep learning techniques that have recently catalyzed dialogue-oriented research [184]. With the improvement of Neural Network methods in dialogs, there has been a surge in the development of task-oriented dialog systems in various domains such as ticket booking [99], online shopping [176] and restaurant searching [166]. Conversational systems that focus on doctor-patient interactions are a significant contribution to the health domain [182, 97, 161].

Over the last decade, it was evident that the dialog systems have shown a significant benefit for health-related purposes [97]. Use of personal conversational agents for healthcare has shown significant improvements in improving the mental health of patients [162, 37], improvements in physical activity [103], improvement in balanced diet [7], and accessibility to online health information [14]. Most of the research on healthcare conversational systems was focused on the monolingual data [97] with a majority of the corpus in the English language. However, in several multilingual regions of the world, it is natural for speakers to produce utterances and responses which are Code-Mixed.

1.1 Code-Mixing Phenomenon

The use of Code-Mixed (CM) utterances is observed in healthcare conversations in a multilingual community. It is observed that physicians often use their language accommodation skills, such as code-mixing, to communicate with their patients in an ethical and supportive manner [170]. The use of code-mixed utterances allows the patient to express their medical condition freely [158]. In the western states, the physicians used African American English (AAE) to connect with the patient when discussing

¹Task-oriented dialog systems can solve a particular task by conversing with the users.

S Utterance

- p good evening sir.
- p sir *ixi* 2 days back *vaccAnu* sir.(I came 2 days before.)
- d *A*.

(Ok.)

p spray koVttukunna waruvAwa oVka nasal block avuwuMxi sir, oVka nasal free avuwuMxi sir.

(After using spray, I am experiencing blockage in one nostril and another nostril is free.)

d *A*.

(Ok.)

p day by day *itla avuwuMxi*.

(This is happening daily.)

d *A*.

(Ok.)

- p *koVMcaM* breathing, 80 percent sneezing *wakva ayiMxi*.(Slightly breathing problem and 80 percent sneezing has decreased.)
- d A axi kUdA sakkagA ayiwaxi.

(It will also reduce.)

p A

(Ok.)

d meVllagA slowgA ayiwaxi.(Gradually, slowly, it will decrease.)

Table 1.1 A Code-Mixed Telugu-English doctor-patient conversation. S, d, p refers to speaker, doctor, and patient, respectively. The Telugu script is written in wx format and italicized. Translations are given in brackets.

personal matters and Standard American English (SAE) to professionally communicate and elicit information regarding medical issues [170]. The Telugu-English Code-Mixed conversation is prevalent in doctor-patient television shows where the doctor interacts with the patient via telecommunications to give diagnostic advice. Such television shows are LifeLine-TV9² and Dr. ETV^3 . Table 1.1 shows an example of the code-mixed doctor-patient interaction obtained from our corpus. When engaging in such conversations, we can see that it is very natural for such speakers to use code-mixed utterances, mixing their native language with English.

CM Sentence:	day by day <i>itla avuwuMxi</i>								
Language:	EN EN EN TE TE								
Gloss:	day by day this happens								
Translation:	This happens day by day								

 Table 1.2 A CM Telugu-English Utterance, with EN being English and TE being Telugu languages.

In multilingual real-life conversations, it is natural for humans to transition between different languages while still maintaining coherent contexts. This transitioning of languages is termed Code-Mixing. Code-Mixing is a well-studied linguistic phenomenon in the field of bilingual and multilingual speech communities. According to [65], Code-Mixing is a "juxtaposition within the same speech exchange of passages of speech belonging to two different grammatical systems or sub-systems. According to [115], Code-Mixing refers to the embedding of linguistic units such as phrases, words, and morphemes of one language into an utterance of another language. An example of CM Telugu-English utterance is given in Table 1.2, "day by day *itla avuwuMxi*". The initial part of the sentence "day by day" is in English, and the end of the sentence, i.e., the italicized part, is in Telugu.

Code-Mixing poses a significant challenge for language technologies due to the intricate fusion of both the languages. Also, one would need to use both syntactic and semantic language processing methods for both the languages simultaneously [8].

Linguists such as [66, 143, 74, 77] theorized that people code-mix/switch for a certain purpose. i.e.,

- 1. Show oneness with another person's community
- 2. Show-off
- 3. Emphasize
- 4. Clarify a particular topic
- 5. Repeat in another code to re-emphasize
- 6. Be fluent

HzZjHtE

²https://www.youtube.com/watch?v=-4NQ8AEtXdE&list=PLlRkq6AzK4ZTPyFBOKNVc6GqD_

³https:/hhhh/www.youtube.com/watch?v=8-fg-VG-el8&list=PLN5na6FPUL3nta_

¹KdsieSiCnr-FHlcFR

7. Maintain a wider vocabulary

According to [147, 114, 65, 58, 12], Code-Mixing often depends on:

- 1. Languages involved the proficiency, familiarity, and cultural significance of the languages for the speakers involved can influence the frequency and extent of code-mixing.
- Geographical locations The geographical location where code-mixing takes place can have an impact. In regions with multilingual communities or areas where different languages coexist, code-mixing is more likely to occur.
- 3. The difference in language interaction code-mixing may be more prevalent in bilingual communities where both languages are used interchangeably in different domains, such as at home, at work, or in social gatherings. Additionally, the purpose and formality of the interaction can also affect the degree of code-mixing.
- 4. Social context The social context in which language is used can impact code-mixing. Different social situations, such as formal or informal settings, family gatherings, workplace environments, or online interactions, may encourage or discourage code-mixing. Social factors like age, education level, social status, and group dynamics can also play a role in determining the occurrence and extent of code-mixing.
- 5. Language proficiency Speakers who are more fluent and comfortable in one language may codemix to compensate for gaps in vocabulary or express complex ideas. In contrast, speakers with high proficiency in multiple languages may code-mix as a form of linguistic creativity or as a way to signal identity or group membership.
- 6. Language dominance In communities where one language is considered prestigious or associated with higher status, code-mixing may occur as a way to incorporate or align with that dominant language while still maintaining elements of the native or heritage language.
- 7. Inter-generational transmission In families or communities where different generations have varying levels of language proficiency, code-mixing can occur as a means of communication and bridging generational gaps. Younger generations may mix languages to accommodate older family members who are more comfortable in their heritage language, while also incorporating elements of the dominant language.
- Language policy Language policies implemented by governments or educational institutions can impact code-mixing. Policies that promote bilingualism or multilingualism may encourage code-mixing as individuals navigate between different languages.

So far, most of the research in code-mixing focused on POS tagging and Language Identification [159, 11, 104], identifying the code-switched points [148, 179]. discourse analysis [149], sentiment

analysis [23, 24] and hate speech detection [15, 25, 29], named entity recognition [2, 124]. A major limitation in the computational processing of code-mixed text is the lack of end-to-end user applications that interact directly with the users in multilingual communities.

A few conversational datasets are proposed to promote work on multilingual and code-mixed dialogs. They are Code-Mixed DSTC2 dataset [10], MultiATIS++ [175], FMTOD datasets [142], A task-oriented Dravidian dataset [86]. Code-Mixed DSTC2 dataset is obtained from the original DSTC2 restaurant conversational corpus [73]. Every utterance from the DSTC2 restaurant reservation dataset is converted to a code-mixed utterance using crowd sourcing, thereby indicating that the data wasn't obtained from real-time code-mixed conversations. Similarly, the ATIS corpus [72] is translated into various languages to construct the Multi ATIS++ corpus. Even the task-oriented Dravidian dialog dataset used English utterances and converted them to multilingual Malayalam and Tamil utterances [86]. Facebook's FMTOD dataset [142] is constructed from real-life utterances but lacks multi-turn conversations. Therefore, there was still a need to develop datasets that focus on real-life code-mixed conversations that support multi-turn scenarios.

In the context of code-mixed healthcare, [147] pointed out that understanding how patients feel if they are open will help to give better care, enable better communication, and better uptake of preventative care. Thus, it motivates us to create datasets containing multilingual and code-mixed conversations and thus facilitate the development of code-mixed conversation systems in the medical domain.

Therefore, we created a code-mixed task-oriented dialog dataset that captured conversations between doctors and patients in Telugu and Dakhini. Telugu and Dakhini are spoken in Telangana and Andhra Pradesh, the two states of South India. The matrix language used in the CM dialog dataset is either in Telugu language or Dakhini. Telugu is the regional language spoken in Telangana and Andhra Pradesh, the two states of South India. Telugu ranks fourth among the Indian languages with the highest number of native speakers in India, with nearly 82 million speakers⁴. It is also the 15th most spoken mother language in the world⁵. It is the most widely spoken member of the Dravidian language family. Dakhini is spoken natively in South India. Dakhini, spoken in the Telangana region, is often heavily influenced by Telugu and Urdu [145]. The contact of these languages with English led to the emergence of code-mixed languages known as Telugu-English and Dakhini-English.

Our CM task-oriented dialog dataset has over 3k medical conversations with intents and slot annotations. The dataset supports code-mixing at inter-sentential, intra-sentential, and even at morphological levels. The conversations in the dataset are multi-turn doctor-patient interactions that are task-oriented, where the task is to diagnose the patient's condition.

⁴https://www.censusindia.gov.in/Census_Data_2001/Census_Data_Online/Language/ Statement1.htm

⁵https://www.thenewsminute.com/article/making-telugu-compulsory-mother-tongues-_ last-stronghold-against-hindi-imposition-73014

1.2 Task-Oriented Dialog System

We observe that now-a-days with the increase in types of medical problems, the specializations in the medical field are increasing. However, it is becoming a difficult task for an ordinary man to determine which doctor to visit based on their symptoms.

Let us understand it with the help of an example: Consider a kid having a medical condition, "rhinitis", a.k.a "common cold". As the child has a common cold, the child's parent might prefer to consult a doctor who specializes in "common cold," i.e., the ENT specialist. It might seem logically correct to consult an ENT doctor for a common cold. However, is it the right choice? Often, there will be a situation that the child cannot express their symptoms. In such a scenario, it is essential to consult a doctor specializing in children's medical care, a.k.a pediatrician. Also, the knowledge about new and updated medication concerning the children is available with the pediatrician. The above example suggests that although the patient knows little about the specialist, their choice was vague. Such scenarios will lead to additional hospital visits and delay in treatment. We can minimize this if we develop a system that can direct patients to a specialist who is well-versed in treating their medical condition. In other words, a system that serves as a medical assistant and guides the patient to a specialist based on the patient's symptoms and demographics. To achieve this, a conversation between a human and a computer system must occur where the patient expresses their symptoms, and the system should output the specialist based on those symptoms. Thus, it becomes essential to build a healthcare dialog system. To be specific, we have concentrated on text-based task-oriented healthcare dialog system where the task is to find a specialist based on the patient's demographics and symptoms.



Figure 1.1 Modular Task-Oriented Dialog System Architecture

The modular approach to Task-oriented Dialog (ToD) addresses the complexity of goal-oriented dialogue by breaking it down into sub-tasks. The solution is a pipeline of independently trained and

executed modules (components): the output of each serves as the input to the next. Our dialogue system can read input text and generate output text. The three core components are natural language understanding (NLU), dialogue manager (DM), and natural language generation (NLG). We have also added a few additional features to our dialog system, such as Hate Speech Detection, Suggestion System, and the Medical Specialist Classification. The architecture of the dialog system is given in Figure 1.1. The components are described below.

1.2.1 Natural Language Understanding

In task-oriented dialog systems, natural language understanding (NLU) is the primary module. NLU identifies the main goals expressed in the user's utterances. It usually encompasses two sub-tasks, namely intent classification and slot filling. Intent classification can be visualized as a classification task to predict the intent behind a speaker's utterance. Slot filling can be seen as a token classification or a sequence labeling task to extract the relevant slots by predicting each token's slot label in an utterance. To build an NLU model, we annotated the utterances with intents and slots. The intent and slot annotation helps to understand the conversation and enables the researchers to develop reliable task-oriented dialog systems.

We then test our corpus on the benchmark Natural language understanding (NLU) models via intent classification (IC) and slot filling (SF) subtasks. The dialogs in the dataset are multi-turn, indicating a conversational flow in a dialog. The intent of the previous utterance may depend on the current utterance. We incorporated contextual information into the intent classification model to address this conditional dependence. The contextual IC model uses contextual signals along with the previous utterances to predict the intent. Also, as the slots can be visualized as chunks in an utterance [172], we contemplate the slot-filling task as a chunk identification problem instead of a sequence-to-sequence task. We train the slot-tag model to predict the coarse-grained slots in the first phase and the complete slot chunks in the second phase.

1.2.2 Hate Speech Detection

Healthcare workers are at greater risk of workplace abuse than most other workers, with nurses and family physicians rated as most at risk of abusive encounters with patients [45]. Types of abusive encounters range in severity from verbal threats to more extreme encounters, such as stalking and physical assault. Such hate speech and physical abuse must be strongly condemned for a better function of health care.

In April 2021, a doctor in a government hospital in Madhya Pradesh resigned on Saturday after some leaders allegedly yelled at him following the death of a COVID patient. A video of the incident, which has since gone viral, shows a group of leaders yelling at a doctor who was in-charge of the COVID ward

at the Government JP Hospital in Bhopal⁶. The incident was sparked by the death of a COVID patient, who was admitted to the trauma ward in a critical condition and died during the treatment.

Statistics show that over 75% of doctors across the country have faced at least some form of violence and that 68.33% of the violence was committed by the patient's attenders /escorts ⁷. But these numbers are an underestimate as all the cases of violence are not reported. The violence includes verbal abuse, telephonic threats, physical assault, murder, and arson. The Indian Medical Association (IMA) said the effects of such violent episodes extend longer than the experience itself, where most doctors report having faced insomnia, depression, anxiety, and an inability to see their patients without any fear of violence. The IMA said it had constantly been requesting the government to enact a central law with an IPC section, fixing the responsible officers to investigate, the time frame for investigation, and punishments to the culprits, as early as possible, without the option for bail. The Government of India enacted the Hospital Protection Ordinance under the epidemic Disease Act 1897⁸ to prevent violence against doctors, hospitals and healthcare workers in the last year with the efforts of the Hon'ble Home Minister. Similarly, the Hospitals and Healthcare Prevention Act were enacted and practiced in 19 states.

Medical Community save lives. They play a central and critical role in improving access to and quality health care for the population. They provide essential services that promote health, prevent diseases, and deliver health care services to individuals, families, and communities based on the primary health care approach. Thus, they must be treated with the utmost respect. To decrease the abuse in the healthcare community, we included a hate speech detection module in our dialog system.

Automatic detection of hate speech is crucial in conversations. Often, modern dialog systems are designed to process and train the models even during the simulation of the conversation. If the hate speech is not addressed in the dialog system, the dialog system will process the hate speech conversations, simulate them, and respond to the users in an offensive manner. This scenario was observed in the Tay bot released by Microsoft. Tay was an artificial intelligence chatbot that Microsoft Corporation released via Twitter on March 23, 2016; it caused subsequent controversy when the bot began to post inflammatory and offensive tweets through its Twitter account, forcing Microsoft to shut down the service only 16 hours after its launch⁹.

In order to prioritize respectful dialogue and combat hate speech, we have implemented an automated hate speech detection module in our dialog system. The automatic detection of hate speech is achieved by the machine learning and deep learning models that were trained on monolingual, multilingual, and code-mixed texts [141, 122, 55]. We took such pre-trained hate speech detection models and used zero-

⁶https://indianexpress.com/article/india/mp-doctor-resignation-cong-mla-covid-_ 7268600/

⁷https://timesofindia.indiatimes.com/india/over-75-of-doctors-have-faced-_

violence-at-work-study-finds/articleshow/47143806.cms

⁸https://rajyasabha.nic.in/rsnew/Committee_site/Committee_File/ReportFile/15/161/

²³⁶_2022_1_12.pdf

⁹https://www.bbc.com/news/technology-35890188

shot transfer learning to identify hate speech and abusive content in each user's utterance. If the user's utterance has hate speech, then necessary action is taken by terminating the session.

During the interaction, our dialog system continuously monitors detect hate speech and abusive content. If, at any point, the hate speech detection module identifies the use of hate speech in a user's utterance, the necessary action is taken to terminate the session immediately.

By incorporating this hate speech detection module, we aim to create a safe and inclusive environment where users can engage in constructive and respectful conversations. This proactive approach helps prevent the dialog system from processing and responding to hate speech, thus ensuring that offensive and harmful content is not perpetuated.

It is crucial to address hate speech in AI systems, as demonstrated by past incidents like the Tay bot released by Microsoft. By actively detecting and addressing hate speech, we strive to maintain a responsible and ethical dialog system that promotes positive interactions and fosters a respectful atmosphere for all users.

1.2.3 Dialog Manager

Modern research systems for task-based dialogue are based on a more sophisticated version of the frame-based architecture called the dialogue-state architecture [5, 83]. In the dialog manager, we keep track of what the user has conveyed and what information is required from the user. The dialogue manager maintains the dialogue's current state (which includes the user's most recent intent and the entire set of slot-filler constraints the user has expressed so far). Using the dialog's state, the dialog manager decides what the system should do or say next for the task to be completed. In our proposed system, we use a simpler frame-based approach to implement the dialog manager. The dialog manager keeps track of the information obtained from the NLU module and uses this information to ask relevant questions to the user until all the required information is obtained. This approach is effective for our system's specific use case, which involves gathering medical information from the user to determine the appropriate medical specialist to consult. Thus, the dialog manager plays a crucial role in the success of the dialog system by effectively managing the conversation and ensuring that all the required information is obtained to achieve the task's goals.

1.2.4 Suggestion System

During doctor-patient conversations, we observed that the user might forget to mention some symptoms. However, collecting all the symptoms in the medical domain is crucial to get reliable results. Even if the user forgets to mention some symptoms, the system must be designed in such a way that it should get all the necessary input from the user. In other words, a suggestion system should suggest the relevant symptoms to the patient/user, allowing them to input all the symptoms. In our work, the suggestion system provides relevant symptoms for a user by collecting symptoms from similar users (collaborating). This technique is mostly used in recommender systems, also known as collaborative filtering [131].

1.2.5 Medical Specialist Classification

The medical specialist classification module is an essential component of the dialogue system as it helps patients to find the right medical specialist. It is the final action performed by the dialog manager of our dialogue system. The dialog manager uses the information collected from the user to determine which medical specialist the patient should visit. This process involves analyzing the symptoms, demographics, and other medical information provided by the patient to make an informed prediction. To achieve this, we integrated a medical specialist classification model in the dialog manager. The classification model is trained on the code-mixed dataset. The training process involves feeding the model with user data along with their corresponding medical specialist labels. The model then learns to identify patterns in the input data that are associated with each medical speciality. Once the model is trained, it can be used to predict the medical specialist.

1.2.6 Natural Language Generation

The NLG interacts with the user in the natural language to extract information from the user or to convey information to the user. Once the dialog manager decides on the following sequence of action, we need to generate a textual response to the user. Natural language generation (NLG) takes information from the dialog manager and generates a textual representation for the user.

Now the question arises on how to convert the dialog manager's decision into a textual format. Again, we take advantage of the dataset, and we use delexicalization [116] to mask the symptoms/slots from the original dataset and extract the templates. We can use those templates while interacting with the user.

In our dialog system, the information interacted with the user can be in monolingual or code-mixed format. But the system should decide the language to respond with the user. We use the Markov rule to see the user's last interaction, predict the language the user spoke, and replicate the response in the expected language.

1.3 Motivation

The following reasons explain the motivation behind creating a code-mixed task-oriented dataset and the dialog system:

 Code-Mixing is a conversational phenomenon in multilingual communities. Thus, conversational systems must understand code-mixing in critical domains. The analysis of code-mixed dialog data and the creation of datasets will aid in developing better code-mixed systems.

- 2. Analyzing the doctor-patient code-mixed data to understand the user's intentions and communicating with them by accommodating to their language will help in better communication with the patients and provide improved care.
- 3. Most dialog datasets are monolingual. Even the recent multilingual datasets are created from the monolingual data [10, 86, 175]. Most of them focus on single utterances [175, 142]. *Ours is the only code-mixed dialog dataset for the medical domain taken from real-life conversations. The real-life dialog datasets can be used to build practical applications, and the models developed on them can handle real-life conversations effectively.*
- 4. Nowadays, it has become increasingly popular for patients to seek advice from online resources before consulting a doctor. The typical online resources where people seek advice are Google, healthcare websites, and chatbots on hospital websites. These online resources, healthcare websites, and chatbots provide detailed information about the symptoms given by the user and diagnose their medical condition using artificial intelligence algorithms. Such online resources offer more than relevant information to the user. Online resources, if used extensively, might lead to a form of medical anxiety in the user. In our view, visiting a medical specialist is most important rather than relying on online resources. We focussed on developing a dialog system that provides information on a specialist based on the patient's symptoms and demographics.

1.4 Challenges of Data collection and Annotation

The challenges of dataset collection and annotation are explained below.

- 1. Even though code-mixed doctor-patient interactions are common in day-to-day life, collecting the data is a strenuous task as it requires approval from both the doctors and patients. There were situations where the doctor or the patient was hesitant to approve the data collection.
- 2. Initially, the conversations were in audio format. To maintain the anonymity of conversations, we opted for speech-to-text processing. The speech-to-text translation was challenging as the Telugu tokens in our corpus were based on the "Telangana" dialect. In contrast, ASR¹⁰ (automatic speech recognition) systems were trained on the "Andhra" dialect. Due to dialectal variations and regional variations in spoken utterances, ASR was not possible on the data. We ought to resort to manual speech-to-text processing followed by a vigorous method of correcting speech-to-text errors.
- 3. Selecting intent annotations that optimally tag the data based on tagging schema developed for both medical and conversational data was a difficult task. We first opted for tagging based on syntactic and semantic information from the intent. However, the number of intents has increased

¹⁰https://cloud.google.com/speech-to-text

exponentially. So we opted for tagging the intent of the utterance based on its purpose and role in the dialog.

4. There were lots of medical terms in the dataset. We consulted doctors to understand the medical terms and selected the optimal slot tagset based on various medical entities.

1.5 Major Contributions

The major contributions of this work are:

- We built a Code-Mixed Medical Task-Oriented Dialog Dataset 'Su-Vaid'. It is the first corpus containing code-mixed medical conversations in Telugu-English and Dakhini-English languages. The dataset contains 3005 Telugu-English Code-Mixed dialogues between patients and doctors with 29k utterances covering ten specializations with an average code-mixing index (CMI) of 33.3%.
- 2. We have annotated the dataset with intents and slot labels. A set of 19 intents and 34 slot labels, are used for annotating the data.
- 3. The dataset supports inter-sentential and intra-sentential code-mixing. As shown in 2^{nd} , 4^{th} and 6^{th} utterances in Table 1.1. code-mixing happens even at the morphological level.
- 4. We analyzed the code-mixed dataset by comparing the medical conversations with the established medical-conversational theories, conversational analysis, and code-mixed linguistic theories.
- 5. We developed a dialog system that predicts the medical specialist based on the patient's symptoms. The dialog system uses a modular approach by breaking down the system into subparts viz. NLU, Hate Speech Detection, DM, Suggestion System, Medical Specialist Classification and NLG.
 - (a) The NLU uses contextual ground truth labels for intent classification and chunk identification for slot filling.
 - (b) A Hate Speech Detection module that is trained on BERT ensembles and meta embeddings is used for detecting abusive comments from the user.
 - (c) Dialog manager uses dialog state to track the current intent and slots, and the dialog manager decides the next sequence of action.
 - (d) We have also implemented a Suggestion System to extract more information from the user.
 - (e) The medical specialist classification provides patients with an appropriate referral to a medical specialist.
 - (f) We used delexicalization to extract templates for the NLG module. We also used style accommodation to accommodate the user's language.

1.6 Organization of the thesis

The thesis is organized as follows:

- 1. Chapter 1 introduces the research statement. It presents the aim of the thesis and puts forth the components of the dialog system. It also elaborates on the motivation, major contributions and challenges of our research.
- 2. Chapter 2 provides a comprehensive review of the existing literature related to code-mixed datasets and code-mixed hate speech detection. The chapter also summarizes the state-of-the-art techniques and methods developed for the individual components, i.e., NLU, DM, NLG, Hate Speech Detection, Suggestion System, and Medical Specialist Classification present in our task-oriented dialog system.
- 3. Chapter 3 provides a detailed description of the proposed code-mixed dataset with annotation schema used for the dataset. The chapter also presents the analysis of data across various medical, conversational, and linguistic theories.
- 4. Chapter 4 explains the proposed Natural Language Understanding (NLU) component of the dialog system. It uses a neural network-based approach to perform intent classification and slot filling. The proposed contextual NLU model is evaluated on the code-mixed dataset and achieves promising results.
- 5. Chapter 5 of the thesis provides a comprehensive explanation of the proposed Hate Speech Detection modules, their implementation, and evaluation on the code-mixed dataset. The modules are designed to address the challenge of detecting hate speech in code-mixed data and provide a crucial component for creating a safe and inclusive environment for all users.
- 6. Chapter 6 explains the frame-based dialog manager and the suggestion system that provides additional symptoms to the user. The dialog manager uses a structured approach to handle the conversation flow. While, the suggestion system is developed using content-based collaborative filtering.
- 7. Chapter 7 presents the medical classification system that takes all the input from the data and predicts the specialist. The system uses mBERT with a sequence classification head. The mBERT is finetuned on the available code-mixed dataset. The system achieves high accuracy and F1 score, making it an effective model for users to find the right specialist for their medical condition.
- 8. Chapter 8 presents the Natural Language Generation (NLG) Module. The module uses predefined response templates and incorporates style accommodation too while generating responses. The NLG module produces responses that are fluent, coherent, and relevant to the user's input.

- 9. Chapter 9 presents the integrated dialog system. The system is developed using the independent units mentioned above. The graphical user interface for the dialog system is also presented. We further compare the performance of the model with ChatGPT.
- 10. Finally, Chapter 10 summarizes the thesis and presents the future scope of this work.

Chapter 2

Literature Survey

This chapter lists the task-oriented dialog datasets. We present the work accomplished in the context of the code-mixed datasets and code-mixed hate speech detection. We also summarize multiple SOTA methods developed for the individual components, i.e., NLU, DM, NLG, Hate Speech Detection, Suggestion System, and Medical Specialist Classification present in our task-oriented dialog system.

2.1 Dialog Datasets

Several large annotated corpora exist for generic conversations and specific tasks. We mention a few notable ToD corpora in this section. They are ATIS [72], an airline travel planning corpus, SNIPS [43], an IoT dialog corpus. Some corpora are designed to span across various domains, such as MultiWOZ 2.0 [20] and CrossWOZ [185]. The datasets mentioned above were available mostly in English. Recently there has been increased attention towards expanding the datasets in multilingual and code-mixed domains. Such as Facebook Multilingual Task-oriented corpus (FMTOD) [142] that spans three languages English, Spanish, and Thai. The original ATIS dataset has been translated into several languages known as MultiATIS++ [175]. It covers nine languages in four different language families, including Indo-European (English, Spanish, German, French, Portuguese, and Hindi), Sino-Tibetan (Chinese), Japonic (Japanese), and Altaic (Turkish). Also, [10] expanded the DSTC2 restaurant reservation dataset to facilitate the understanding and development of code-mixed dialog systems. Due to its promising potential, research about task-oriented dialog systems has attracted increasing attention in the medical domain. [161] constructed a dataset collected from the pediatric department in a Chinese online healthcare community. To our knowledge, we found no annotated multi-turn datasets that support code-mixing in the healthcare domain.

2.2 Code-Mixing in Natural Language Processing

Code-mixing is a phenomenon that has attracted significant attention in linguistics. It refers to using two or more languages or language varieties within a single communication event. This phenomenon is

often observed in multilingual communities and has become increasingly common in today's globalized world.

Research articles have been published on code-mixing, exploring its linguistic, sociolinguistic, and psycholinguistic aspects. One such paper is "A Survey of Code-switching: Linguistic and Social Perspectives for Language Technologies" [47]. This paper focuses on the challenges of identifying the languages used in code-mixing scenarios, which is essential for natural language processing applications. This paper also highlights the research conducted on code-switching (C-S) in India, highlighting the multilingual nature of the country and the influences of various languages on Indian languages. The studies cover a range of contexts, including daily life interactions, movies, TV shows, advertising, and immigrant contexts. The studies show how C-S is used to establish identity, social dynamics between characters, and the socio-cultural context of the movie. The research also emphasizes the importance of studying C-S in multilingual contexts and how small sets of data collected through personal interactions can provide crucial information about a region's social and cultural dynamics.

The paper [147] provides an overview of the challenges and techniques involved in processing codeswitched speech and language, such as the lack of labeled data, the complexity of linguistic structures, and the variability in code-switching patterns. The paper also provides an overview of existing codeswitched speech and language processing techniques, including rule-based and data-driven approaches. Furthermore, the paper discusses various applications of code-switched speech and language processing, such as speech recognition, machine translation, and sentiment analysis. Finally, the paper concludes with future research directions and the potential impact of code-switched speech and language processing on multilingual communities. Overall, the paper provides a comprehensive survey of code-switched speech and language processing, highlighting the challenges, techniques, and applications in this rapidly growing field.

There has been significant interest in building language technologies for code-switched languages over the last few years, spanning a diverse range of tasks. Such as,

- 1. Language Identification: Code-switching poses a challenge for language identification systems as they need to determine the language of each word in a code-mixed sentence. The research included using a language model and token classification methods to assign a word to a particular language [104, 109, 54].
- 2. Part of Speech Tagging: Code-switching can affect the accuracy of part of speech tagging, as different languages may have different tagging schemes. Some approaches use language-specific taggers for each language and then combine the results. At the same time, others used a single model that handles code-switching. Other researchers studied joint learning and deep learning models to address language identification and POS tagging in CS tasks [4, 9, 54].
- 3. Sentiment Analysis: Sentiment analysis of the code-switched text is challenging as sentiment words and phrases may not exist in both languages. Some approaches combine the results with

language-specific sentiment lexicons, while others use multilingual embeddings to represent sentiment across languages [49, 51, 30, 28, 31, 22, 70].

- 4. Hate Speech Detection: Code-switched text can also pose challenges for hate speech detection, as hate speech is often implicit and/or sarcastic. Most of the HSD research used language-independent models that can handle code-switching [30, 70, 29].
- 5. Named Entity Recognition: Code-switching can also affect named entity recognition, as named entities may exist in one language but not in the other. Some approaches use bilingual or multilingual resources, while others use cross-lingual transfer learning to improve performance[124, 168].
- 6. Code-mixing presents a challenge in machine translation as well. Machine translation systems often assume that the source and target languages are distinct and have different rules and structures. However, this assumption is often violated in code-mixing scenarios, making it challenging to train machine translation models. The primarily used approach is to treat code-mixed data as a separate language and train machine translation models for this language. This approach has been used for specific code-mixed varieties such as Hinglish (a mixture of Hindi and English), Spanglish (a mixture of Spanish and English)[50, 150]. [34] presents the findings on machine translation for code-mixed data.
- Dialog Systems: Code-switching is often regarded as a conversational phenomenon in multilingual dialog systems. This domain is relatively new and has yet to be researched to a great extent. The current approaches use deep learning language models that can handle multiple languages at once [10, 86, 68].

The papers [89] and [3] proposed the first evaluation benchmarks for code-switching that spans multiple tasks in multiple language pairs. The GLUECoS benchmark [89] consists of the following code-mixing tasks in Spanglish and Hinglish for Language Identification (LID), Part of Speech (POS) tagging, Named Entity Recognition (NER), Sentiment Analysis, Question Answering and Natural Language Inference (NLI) tasks. The LINCE benchmark [3] covers Language Identification, Named Entity Recognition, Part-of-Speech Tagging and Sentiment Analysis in four language pairs: Spanglish, Nepali-English, Hinglish, and Modern Standard Arabic-Egyptian Arabic.

In the Indian language context, Hindi-English is the most widely studied language pair for computational processing, with some recent work on Telugu-English, Tamil-English, Malayalam-English, Kannada-English, Bengali-English, and Gujarati-English. However, language technologies lack conversational applications that interact directly with users in multilingual/code-mixed communities. Our work is the first attempt to computationally understand the code-mixed real-life medical utterances in a multi-turn scenario.

2.3 Code-Mixed Hate Speech Detection

The paper [128] conducted a comparative study of different state-of-the-art hate speech detection methods in Hindi-English code-mixed data. The authors have collected the Hindi-English code-mixed datasets from the HASOC 2019 shared task [106] and from papers [16, 93]. It was the first survey on Hindi-English code-mixed data sets. The dataset contains posts/tweets written in both the Roman and the native Devanagari script. They analyzed the performance of the dataset on five different baseline classifiers.

The paper [55] surveyed recent neural-based models that are implemented for hate speech detection in the context of code-mixed datasets. We examined seven code-mixed datasets developed in 4 Indian code-mixed languages Hindi-English, Tamil-English, Malayalam-English, and Kannada-English. The deep learning methods used in the survey consisted of Bi-LSTM (Bi-directional Long Short Term Memory), Attention-Based Bi-LSTM, Convolutional Neural Networks, Transformers, and pre-trained BERT models and their cross-lingual variants. A few methods used pre-processing steps on the code-mixed dataset. And a few models used hate speech lexicon sources. The models tailored to processing the code-mixed data and hate speech lexicon sources performed relatively better than the others. Empirical results have shown that pre-trained multilingual transformer models with selective translation and transliteration achieved the best results compared to other models.

Shared tasks such as Dravidian Code-Mix [31], and DravidianLangTech [26] were organized to identify the offensive language from comments/posts in code-mixed Dravidian Languages (Tamil-English, Kannada-English, and Malayalam-English) collected from social media. Each comment/post is annotated with the offensive language label at the comment/post level. The data has been collected from YouTube comments and Tweets. The top models used deep learning models like LSTM, CNN, and BERT [53, 52] for hate speech and offensive content detection on the given Dravidian data.

2.4 Natural Language Understanding

There has been much progress towards building task-oriented NLU systems over the past two decades [163, 101]. The traditional approaches for intent classification used machine learning classifiers such as support vector machines (SVM), and logistic regression (LR) models [69, 32]. Whereas the Hidden Markov models (HMM) [160] and conditional random fields (CRF) [129] were used for slot-filling tasks. With the advances in deep learning, NN models have achieved state-of-the-art performance for a wide range of NLP tasks. Various architectures include Elman Recurrent Neural Networks (RNN) [177], Convolutional Neural Networks (CNN) [174], Encoder-Decoder models with attention mechanism [100] and transformer models [33] for NLU tasks.

The most recent approaches viewed that intents and slots in an utterance are correlated. Thus, the joint NN architectures were developed to capture the correlations between the two tasks [100, 63, 33, 172]. [13] investigated the importance of context in task-oriented dialogs for NLU tasks using HMM's

and CRF models and proved that modeling contextual information improved the model's performance for the intent prediction task.

Later on, the work progressed towards the use of NN models in contextual modeling, and the attention-based contextual models were developed to model long-term knowledge transfer for multi-turn dialogs [35, 185]. One such contextual model used previous intent and slot-tagged information as contextual signals to predict the intent and slot labels for the current turn [67]. The paper proved that adding such contextual cues improved the accuracy of the intent classification tasks.

On the other hand, cross-lingual transfer learning models [175] and cross-lingual contextual word representations [142] investigated multilingualism in utterances. The authors [175] have presented that multilingual BERT substantially improves multilingual training and cross-lingual transfer tasks. Whereas cross-lingual embeddings have shown a competitive performance w.r.t transfer learning multilingual models. However, the multilingual scenario is analyzed only for single-turn utterances. The paper also presents a transfer learning NN with contextual modeling and slot chunk detection of multi-turn code-mixed dialogs. So far, no work has been done toward computationally understanding the real-life code-mixed multi-turn utterances via contextual intent classification and slot-filling subtasks.

2.5 Dialog Management

Dialog management is the process of tracking and controlling the conversion to direct the user toward a specific task. Dialog is the major component of a spoken dialog system. It interacts with various components in the dialog system and external knowledge sources. It generally controls the dialogue flow and directs the user toward the task.

The dialog manager can be viewed as two main tasks:

- 1. Dialog state, it tracks the state of the dialog
- 2. Dialog control makes decisions about the next system action.

The dialog state stores a record of what has been said so far in the dialog. It provides the information used for dialog control. Dialog control describes the information to be gathered in the dialog. It holds the information about one or more slots to be filled with values provided in the course of the dialogue.

There are two types of dialog managers: rule-based and Statistical dialog managers.

Rule-based dialogue management is still the most popular solution for industrial task-oriented dialogue systems for their interpretability. Finite-state (FS)-based DM [62, 110], one of the typical rulebased DM, is the simplest and most widely-used method to date. FS-based DM component converts the finite state automata (FSA) based on the output of the NLU and generates dialog act based on this output; such is achieved by manually designing an FSA to analyze a conversation scenario. The [185]¹ uses Rule-Based Dialog state tracking, RuleDST takes as input the previous system state, and the last user dialogue acts. Then, the system state is updated according to hand-crafted rules.

¹The first large-scale Chinese Cross-Domain Wizard-of-Oz task-oriented dataset
In recent years, statistical dialog managers that are data driven, have been proposed for task-oriented dialogue. The most representative traditional DM is the neural belief tracker (NBT) proposed by scholars from Cambridge University in 2017 [113]. NBT uses neural networks to track the state of complex dialogs in a single domain. By using representation learning, NBT encodes system actions in the previous round, user statements in the current round, and candidate slot-value pairs to calculate semantic similarity in a high dimensional space and detect the slot value output by the user in the current round. Therefore, NBT can identify slot values that are not in the training set but semantically similar to those in the set by using the word vector expression of the slot-value pair. The other data-driver approaches for dialog managers used end-to-end architectures. These models simulated the working of dialog managers implicitly. [171] propose a task-oriented dialogue BERT (ToD-BERT), which is trained on 9 English task-oriented datasets across over 60 domains and achieves good performance on four dialogue subtasks. The data-driven dialogue systems, usually with end-to-end structures, are easier to deal with complex conversations. Still, such methods require plenty of training data, and the behaviors are less interpretable.

As the statistical approaches often require plenty of training data and deep annotation for the dialog manager. Also, Their behavior is less interpretable. So we opted for a more reliable and robust framebased dialog manager.

2.6 Recommender System

Our suggestion system, which suggests symptoms to the patients by comparing the symptoms of similar patients, is based on recommender systems. Recommender systems (RS) use different sources of information to provide users with predictions and recommendations of items. Recommender Systems collects information on the user's preferences for a set of items (e.g., user's rating on movies, songs, books, jokes, gadgets, applications, websites, travel destinations, and e-learning material). The information can be acquired explicitly (by collecting users' ratings) or implicitly (by monitoring users' behavior, such as movies or videos watched, songs liked/heard, applications downloaded, websites visited, and books read). Recommender systems may use demographic features of users (like age, nationality, and gender). Also, Social information, like followers, influencers followed, tweets, comments, and posts. There is a growing trend toward the use of information from the Internet of things (e.g., GPS locations, RFID, real-time health signals)

The internal functions for RS are characterized by the filtering algorithm. The most widely used classification divides the filtering algorithms into (a) collaborative filtering and (c) content-based filtering among others. Content-based filtering [96], extracts features from the input text and makes recommendations based on the choices made by the user in the past. Collaborative filtering [17] is based on the assumption that people who agreed in the past will agree in the future. Thus, we can make recommendations to each user based on information provided by those users we consider to have the most in common with them. Of all the methods, collaborative filtering is the most frequently used algorithm for recommender systems. Collaborative filtering provides strong predictive power for recommender systems and requires the least information at the same time. This model can be classified into Memory-Based and Model-Based Collaborative Filtering methods. Memory-Based Collaborative Filtering [1] can be further divided into User-Based and Item-Based Collaborative Filtering methods. User-Based Collaborative Filtering compares similarities between users and recommends a list of top N items based on the rating of each item of a similar user group. Item-Based Collaborative Filtering predicts an item by using the similarity between the item and the item selected by the user. Memory-Based Collaborative Filtering uses technologies such as Vector Cosine Correlation, and KNN to create similar groups among users and recommend items to users within the same group. Model-based methods [1] use recommender system information to create a model that generates the recommendations. The most widely used models for the model-based recommender system are Bayesian classifiers [118], neural networks [183], and Latent Semantic Indexing (via singular value decomposition (SVD)), among others.

Because of the importance of prediction in different genres (such as music, games, videos, and online inventory), there has been huge research on recommender systems. A few notable models are presented here.

The most widely used algorithm for collaborative filtering is the k Nearest Neighbors (kNN) [1]. In the user-to-user version, kNN executes the following three tasks to generate recommendations for an active user: (1) determine k users neighbors (neighborhood) for the active user a; (2) implement an aggregation approach with the ratings for the neighborhood in items not rated by a; and (3) then select the top N recommendations.

The Singular Value Decomposition (SVD) has been generally used as a dimensionality reduction technique in machine learning. In the context of the recommender system, the SVD is used as a collaborative filtering technique. The paper [138] talks about using the SVD for collaborative filtering. The paper presents that the mapping facilitated a clear representation of relationships between users and items.

The paper [183] presented the survey of deep learning techniques for recommender systems. The neural networks covered in the model are MLP, RNNs, CNNs, Auto Encoder, Attention Models, Restricted Boltzmann Machine (RBM), Neural Autoregressive Distribution Estimation (NADE), and Neural Autoregressive Distribution Estimation (NADE). The paper pointed out that, Contrary to linear models, deep neural networks are capable of modeling the non-linearity in data with nonlinear activations such as ReLU, sigmoid, tanh, etc. This property makes it possible to capture complex and intricate user-item interaction patterns. They covered topics of Hashtags recommendation, news articles recommendation, Review text recommendation, and quote recommendations in their survey on text-based recommender systems.

The deep bidirectional self-attention-based transformer architecture has achieved tremendous success in the field of language understanding. In the paper [155], the authors introduce a deep bidirectional sequential model called BERT4Rec for the sequential recommendation. They introduced the Cloze task for model training, which predicts the masked items using both left and right contexts. Exten-

sive experimental results on four real-world datasets show that the model outperforms several NN-based state-of-the-art baselines.

2.7 Medical Specialist Classification

The medical specialist classification model tries to automatically predict the appropriate specialty based on the symptoms and medical information that is given by the patients. [107] presents a meta embedding based classification model. The dataset consisted of more than 40,000 items, including eight departments, such as Otorhinolaryngology, Pediatrics, and other common departments. The experiment results show that the model achieves more than 93.5% accuracy and has a high generalization capacity when compared to machine learning approaches.

The paper [102] has studied doctor recommendations in online health forums. They have trained a BERT model to learn the relationship between doctor and patient embeddings and used a series of encoders and multi-layer perceptrons (MLP) to recommend the correct doctor based on the patient's symptoms.

The paper [130] work provides a predictive model for selecting the most appropriate medical specialists to diagnose a patient. The models used were logistic regression (LR), decision trees, and MLPs. The dataset consists of 58177 patients consulted by 11 different medical specialists. The top-performing LR model achieved 88% accuracy in predicting the specialists.

2.8 Natural Language Generation

Natural language generation transforms a structured dialogue act into a natural language sentence. The following are recent works in the field of NLG.

The paper [185] provided a template-based model (named TemplateNLG) for natural language generation. For TemplateNLG, they extracted templates from the training set and manually added some templates for infrequent dialogue acts.

Encoder decoder models generally do the mapping from frames to delexicalized sentences [164, 165], trained on large hand-labeled corpora of task-oriented dialogue.

The paper [48], presents a survey on natural language generation. This survey gives the latest synthesis of deep learning research on the NLG core tasks, as well as the architectures adopted in the field. The survey concentrates on Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), transformers, Graph Adversarial Networks (GAN), and Copy and Pointing Mechanisms, Memory Networks, and Graph Neural Networks (GNN). They analyzed the datasets developed for Text Summarization, Question Generation, Response Generation, and Distractor Generation.

2.9 Summary

In this chapter, we presented the literature survey on the creation of dialog datasets and the related work in the field of code-mixing for natural language processing and hate speech detection. We also presented multiple SOTA methods developed for the individual components, i.e., NLU, DM, NLG, Hate Speech Detection, Suggestion System, and Medical Specialist Classification present in our task-oriented dialog system.

Chapter 3

A Code-Mixed Task-Oriented Dialog Dataset for Medical Domain

In this chapter, we present the data acquisition of Su-Vaid. The transcription of the Su-Vaid dataset from speech to text. The percentage of code-mixing in the dataset. The analysis of dataset w.r.t various medical, conversational, and linguistic theories And intent, and slot annotation of data for NLU task and annotation analysis.

3.1 Data Collection

To enable the research and development of multilingual and code-mixed dialog systems in the healthcare domain, we provide a code-mixed medical task-oriented dialog dataset. The process of data collection, data conversion, data annotation, data analysis, and modeling is given in Figure 3.1





Our dataset contains 3005 code-mixed dialogs (consultations) between patients and doctors. The number of specializations considered is 10. The total number of utterances is 29294. Most consultations

#doctors	10
#dialogs	3005
#utterances	29294
#tokens	121475
Avg # of utterances in a dialog	9.77
Max # of utterances in a dialog	68
Min # of utterances in a dialog	2
Avg # of tokens in an utterance	4.14
Max # of tokens in an utterance	71
Min # of tokens in an utterance	1

Table 3.1 Statistics of our dataset



Figure 3.2 The distribution of dialogs with respect to medical specialists in the dataset.

start with greetings, and a narration of the patient's medical condition, followed by a multi-turn conversation between the patient and doctor. The doctor tries to gather information regarding the duration of the current disease, medical history, medications, and attributes required for the medical diagnosis from the patient. Optionally, at the end of the consult, the doctor makes a diagnosis and treatment suggestions to the patient. Data acquisition occurred at various medical consultation centers in Warangal, Telangana, India. Specialists belonging to Dermatology, General Surgery, Otolaryngology, Paediatrics, Urology, General Medicine, Gynaecology, Orthopaedics, and Opthalmology, and thousands of patients have volunteered for the data collection. The doctor-patient privileges are maintained during data collection. Manual pseudonymization is followed. In manual pseudonymization, pseudonyms replace a personally identifiable doctor and patient information fields within a conversation record. This feature prevents hackers from knowing that the data belongs to a particular patient (separating data from real attributes). We have manually replaced the named entities related to doctors and patients and their personal information with pseudonyms during speech-to-text transcription.

Our doctor-patient interactions contain audio recordings in code-mixed Telugu-English and Dakhini-English languages. Most conversations were in Telugu and Dakhini (regional languages used in the area). The other languages that the speakers are familiar with are Hindi and English.

Proportion of languages in corpus	
At dialog level	
Telugu-English code-mixed dialogs	85.1%
Telugu-Dakhini-English code-mixed dialogs	11.4%
Dakhini-English code-mixed dialogs	3.5%
At utterance level	
Telugu-English code-mixed utterances	47.2%
Telugu utterances	29.1%
English utterances	16.5%
Dakhini utterances	4.3%
Dakhini-English code-mixed utterances	2.3%
Telugu-Dakhini code-mixed utterances	0.5%
Telugu-Dakhini-English code-mixed utterances	0.2%
At word/token level	
Telugu tokens	65.2%
English tokens	25.8%
Dakhini tokens	6.9%

Table 3.2 The proportion of languages in the corpus at dialog, utterance, and word/token levels.

3.2 Data Conversion

After data collection, we manually converted the data from audio to text format, where all the data is scripted in their respective languages.

We transcribed the data from speech to text to ensure the anonymization of doctor and patient conversations. During the speech to text translation, we have manually altered the personally identifiable information (PII) from the patient data to ensure the privacy. The PII includes the elements of patient data that can reveal the patient's identity such as names, addresses, phone numbers, personal id's etc. When the PII's are manually identified, we have replaced it with pseudonyms. For example, we have removed the surnames/last names and altered first names. The specific address of the patient are left out from the conversation, and only the district/city names are considered, this makes the data less specific. The patient id's and other information is left out from the conversation. Only the altered first name of the patient, city/district names as address and age of the patient is used in our dataset thus safeguarding the patient's data.

Initially, we used Google cloud's Speech-to-Text¹ for automated transcription in 2019. We have tested the speech-to-Text application and observed that for a 5min interaction, it could only generate 10% of data with a lot of inaccurate information. The other 90% of the data was completely missing. The observation made from the faulty transcription is: (i). The ASR does not capture code-mixing. (ii). The speech-to-text translation was challenging as the corpus was based on the" Telangana" dialect. In contrast, ASR (automatic speech recognition) systems were trained in the" Andhra" dialect. Due to dialectal variations and regional variations in spoken utterances, we could rely on the ASR for data transcription.

While transcribing from speech to text, the Brahmi script is used if the audio is present in the Telugu language. For Dakhini, Devanagari script is used, and for English, Roman script is used. We have selected a few proficient Telugu-English and Hindi-English bilinguals to transcribe the data. During the speech-to-text conversion, we followed the transcript 1 format². The silences, such as data pause, gap, and lapse, were considered for turn construction. In the turn-taking mechanism, the pause is a period of silence within a speaker's turn, the gap is a period of silence between turns, and lapse is a period of silence when no sequence is in progress, i.e., the current speaker stops talking [83]. We have identified the utterance as an uninterrupted chain of spoken language, while in transcription, each new line consists of a single utterance. Thus, following the above constraints, all the vocal data is converted to the textual format. The dataset details are given in Tables 3.1 and 3.2. The distribution of specialists in each dialog is given in figure 3.2.

To our best knowledge, this is the first code-mixed medical dialog dataset. The dataset has the following advantages.

1. It has 29k utterances with 3k dialogs and covers ten specializations.

¹https://cloud.google.com/speech-to-text

²https://learn.lboro.ac.uk/ludata/cx/ca-tutorials/trans1.htm

- 2. Code-mixing is practiced in three languages. Telugu, Dakhini, and English. There are instances in the dataset where all speakers use three languages in a single utterance.
- 3. We have considered patients belonging to different ages, from small children to the elderly, gender (male and female), occupation, and education for our data collection.
- 4. Nearly half of the utterances in the given dataset are Telugu-English code-mixed, followed by Telugu and English utterances. This denotes the level of the code-mixing present in our corpus.
- 5. Based on the proportion of words in our dataset, the Telugu words are of high frequency in our dataset, which denotes the inclination of our dataset toward the regional language.

3.3 Code-Mixing Index

When analyzing the code-mixed dataset, it is important to quantify the level of code-mixing between languages. At the utterance level, the code-mixing index(CMI) [60] is given by,

$$CMI(u) = \frac{\sum_{i=1}^{n} w_i - \max_{L_i \in \mathcal{L}} (w_{L_i})}{\sum_{i=1}^{n} w_i} \times 100$$
(3.1)

Where u is an utterance in the corpus, \mathcal{L} is the set of all languages in the corpus, w_{L_i} is the number of tokens in a language L_i in an utterance. $\max_{L_i \in \mathcal{L}} (w_{L_i})$ is the maximum number of tokens of a language L_i in an utterance, and $\sum_{i=1}^n w_i$ is the number of tokens in an utterance (excluding named entities as they are language independent). The formula $\max_{L_i \in \mathcal{L}} (w_{L_i})$ is inclined towards counting the number of tokens in the matrix language³. Thus, Equation 3.1 gives the percentage of foreign language tokens in an utterance. For monolingual utterances, we will get CMI = 0 as $\max(w_{L_i})$ will be equal to $\sum_{i=1}^n w_i$.

Code-mixing on the dataset is calculated by taking the average of the CMI across all sentences in the dataset. Given by

$$CMI_{avg} = \frac{1}{N} \sum_{i=1}^{N} CMI(u_i)$$
(3.2)

N is the total number of utterances in our data. Our dataset's average code-mixing index (CMI_{avg}) is 33.3%. Our dataset has a relatively high amount of code-mixing compared to the code-mixed datasets given in [60].

3.4 Data Analysis

We analyzed our data by comparing our corpus with the relevant medical, conversational, and linguistic theories.

³The dominant language in the code-mixed utterance, into which elements of the second language are embedded.

- 1. Comparing our corpus w.r.t medical conversational theories will help analyze our data's relativeness to the medical sector.
- 2. The conversational analysis will help in building effective dialog-oriented applications.
- 3. Code-mixed linguistic analysis helps in language understanding and promotes ease of natural language processing from a multilingual perspective.

3.4.1 Medical Conceptual Analysis

S	Utterance
d	eVMwa vayasu?
	(How old are you?)
p	45.
d	emEMxi?
	(what happened?)
p	muKaM gokudu ayiwuMxi sir. cewula mIxa, kAlYla mIxa, meVda mIxa, bAgA avasWa
	ayiwuMxi.
	(There is itching on face, sir. On legs, on the neck, it is troublesome.)
d	eVMdalo wiruguwavA?
	(Do you roam when it is sunny?)
p	avunu sir.
	(yes sir)

 Table 3.3 A doctor-patient conversation

This section analyzes our corpus with the hypothesis established on medical conversations.

The survey on doctor-patient conversations by [75] has uncovered some of the crucial work in this field. They have pointed out that most of the medical conversations were doctor-centered. We have observed the same pattern in our data where the doctor directed the flow of conversation. An example is shown in Table 3.3.

In some of the patient and doctor conversations, a few doctors exhibited patient-centered care. In patient-centered care, the doctors have shown keen listening and compassion toward patients' conversations. Such interest in the patient improved the patient's condition psychologically and physiologically. Such patients were more interested in revisiting the same doctor if there was any need for further medical consultations and treatments. In our corpus, most of the statements are followed by acknowledgments from doctors. It can be seen in the conversations in Tables 3.3, 3.4, and 3.5.

S Utterance

- p Doctor gAru oVka kaMtiki glaucoma vaswe maro kaMtiki vacce avakASAlu unnAyA?
 (Doctor, is it possible to get glaucoma from one eye to the other?)
- d Glaucoma *anexi* bilateral disease.

(Glaucoma is bilateral disease.)

- d Bilateral disease *aMte reVMdu prakkalaki vacce jabbu*. (Bilateral disease is a disease that affects both sides.)
- d *kAnI* by nature *reVMdu kalYlYaki oVkesAri rAxu axi oVka rakaMgA* blessing in disguise *ani anukovaccu manaM*.

(But by nature, it does not come to both eyes at once, we can think of it as a kind of a blessing in disguise.)

Table 3.4 A doctor-patient conversation

We observed that the patient inquired about the medical condition, and the doctor explained it to him as given in the conversations in Table 3.4. The related studies by [18, 151] showed that eliciting the patient's view of the illness increased understanding and commitment to following the doctor's advice.

In some conversations, the patients could not speak about their medical condition because of modest feelings [154]. Nevertheless, when the doctor explained the problem using medical terminology and medical investigations, it helped the patient look at the problem from a medical perspective, and they spoke more about the medical condition.

[167] showed that most of the patients asked fewer questions than physicians and were less likely to receive answers to them than physicians were. This scenario was observed in our interactions. The doctors just remained calm instead of replying to the patient's inquiries. However, the patients participated in communication whenever the doctor asked questions.

3.4.2 Conversational Analysis

This section analyzes our corpus based on the existing conversational theories.

We can view doctor-patient interactions as a domain-dependent task-oriented conversation with the domain being medical and the task being diagnosis and treatment.

Doctor-patient interactions begin with an opening sequence, progressing through problem presentation, history taking, physical examination, diagnosis, and treatment recommendations, and then on to a closing sequence [21, 75, 132]. (Other types of medical visits-follow-up, routine maintenance, and physical check-ups have distinctive but equally patterned structures.). From the doctor's point of view, this structure is taught in medical school, and patients are repeatedly trained by exposure to it from childhood [152].

We have witnessed that the CA studies of ordinary conversation, such as turn-taking [135] and repair mechanisms [139] are largely followed in our clinical interactions.

Grounding

It is important for the participants to establish what they both agree on, called the common ground. Speakers do this by grounding each other's utterances. Grounding means acknowledging that the hearer has understood the speaker. Grounding is also observed in doctor-patient utterances. The doctors constantly ground each other's utterances by explicitly saying "ok" or repeating what the patient said. It can be seen in doctor-patient conversations in Table 3.5.

Sub dialogues and Dialogue Structure

Medical conversations have a structure. A few examples are shown in Table 3.5. We can see that the doctor's question is often followed by an answer from the patient. The medical statements from patients are mostly followed by acknowledgments from the doctors. The doctor's query related to any symptoms is followed by either confirmation or rejection from the patient. The patient's query about the medical condition is mostly followed by a medical explanation from the doctor. Compliments from doctors often gave rise to downplaying from patients or vice-versa. These pairs in the conversational analysis are called adjacency pairs. These adjacency pairs help to develop a better predictive analysis in goal-oriented dialog systems.

Sub Dialogs	S	Utterance	Intent
1 (Paediatrics)	d	emannA medicines vAdArA ammA I 3, 4	INFO_REQUEST
		yearslo bAbuki?	
		(Did you use any medications for the boy?)	
	р	lexaMdI.	ANSWER
		(No.)	
2 (Otolaryngology)	d	eVnni rojula nuMci?	INFO_REQUEST
		(Since how many days?)	
	р	15, 20 days ayiwAMxi.	ANSWER
		(Its been 15, 20 days.)	
3 (Dermatology)	d	blood test emEnA ceyiMcArA aMdi?	INFO_REQUEST
		(Did you do any blood test?)	
	р	A ceyiMcAnu aMdi oVkasAri.	ANSWER
		(Yes, i had done it one time.)	
4 (Opthalmology)	d	emEMxi ceVppammA?	INFO_REQUEST
		(Let me know what happened?)	

Sub Dialogs	S	Utterance	Intent
	р	eVdama kannu xuraxa peVduwuMxi.	ANSWER
		(Left eye is itchy.)	
5 (Gynecology)	d	Periods regular <i>levA</i> ?	INFO_REQUEST
		(Are periods not regular?)	
	р	levu —	ANSWER
		(No.)	
5 (Dermatology)	р	hair fall <i>uMxi</i> madam.	STATEMENT
		(I have hair fall.)	
	d	ok.	ACK
6 (Otolaryngology)	р	sir nenu narAla balahInawa tabletlu eVp-	STATEMENT
		patikI vesukuMtA sir.	
		(I always use nerve weakness tablets (a.k.a	
		multivitamins)).	
	d	ok.	ACK
7 (General Medicine)	р	TSH vaccesi 5.95 uMxi.	STATEMENT
		(TSH is 5.95.)	
	d	hmm.	ACK
8 (General Surgeon)	р	bAgA constipation vaswuMxi, bAgA bleeding	STATEMENT
		avuwuMxi.	
		(I am having heavy constipation and heavy	
		bleeding.)	
	d	ok.	ACK
9 (Urology)	d	malYlYi scan wIswe weVluswaxi kidney	STATEMENT
		stones uMxo lexo ani.	
		(We will get to know if kidney stones are	
		present or not after the scan is done.)	
	р	sare sir.	ACK
		(Ok sir.)	

Table 3.5 Sub dialogs in the dataset. S refers to the speaker of the data, d refers to the doctor's utterance and p refers to the patient's utterance.

Initiative

In the doctor and patient interaction, the doctor mostly controls the conversation as the doctor is responsible for the diagnosis and treatment of the patient. If the patient has questions regarding the diagnosis, treatment, or medication, the patient has the initiative. We observed that the doctor took the primary initiative in most of the conversations.

Implicature

It is the action of implying a meaning beyond the literal sense of what is explicitly stated. We observed the following implicatures in all the doctor and patient specializations. In dermatology, the utterance "*sUxi guccinatlu noVppi uMxA*? (Is there piercing pain?)" is implied to measure the degree of the patient's pain in the affected area. In ophthalmology, the utterance "paper *caxuvoVswuMxA*? (Are you able to read the newspaper?)" implied that "Is your near vision correct?"

3.5 Role of code-mixing in our dataset

Linguists like [66, 143, 74, 77] theorized that people code-mix/switch for a certain purpose. i.e.,

- 1. Show oneness with another person's community
- 2. Show-off
- 3. Emphasize
- 4. Clarify a particular topic
- 5. Repeat in another code to re-emphasize
- 6. Be fluent
- 7. Maintain a wider vocabulary

In doctor-patient interactions, we observed the following factors that motivated the speakers to use code-mixed utterances. They are

- 1. The doctors used code-mixing for fluency, conversational flow, grounding, and to express empathy.
- 2. Code-mixing helped the patients to emphasize their medical condition/symptoms.
- 3. The presence of a wide vocabulary helped doctors and patients to communicate fluently.

Consider an utterance from our dataset. For the utterance "day by day *itla avuwuMxi*", there also exists the monolingual counterpart, which is also used in formal and informal conversations, "*prawi roju ilA avuwuMxi*" (English translation: "It happens daily"), Even though the monolingual sentence is trivial but the speakers didn't use it in during their conversation. We also cite some code-mixed utterances and their monolingual counterparts from our corpus in Table 3.6.

From the utterances given in Table 3.6, We can see that the speakers preferred code-mixing instead of these trivial monolingual sentences. As mentioned above, the factors such as fluency, emphasis, maintaining a conversational flow, and easy understanding motivated the speaker to use code-mixed text in our doctor-patient interactions.

code-mixed utterances	Monolingual Counterparts	English Translation
hair fall bAgA ayiwuMxi	juttu bAgA rAluwuMxi	Heavy hair fall
ilA face mIxa nallati maccalu	ilA muKaM mIxa nallati mac-	I am having black spots
vaswunnAyi	calu vaswunnAyi	on my face
morning ayiwuMxA? night ayi-	poVxxuna avuwuMxA? rAwri	Will it be in the morning?
wuMxA?	avuwuMxA?	Will it be at night?
neck pain vaswoMxi sir	meVda noVppi vaswoMxi sir	I am having neck pain sir
actualgA nA age vaccesi 25	vAswavAniki nA vayassu 25	Actually my age is 25
I was having like, oVka 4 years	nAku 4 saMvawsarAla kriwaM	I had varicose veins prob-
kiMxa nAku varicose veins prob-	anArogya siralu uMdevi	lem 4 years ago
lem uMde		

 Table 3.6 Examples of code-mixed utterances taken from the corpus with their monolingual counterparts.

Linguistic Analysis

A few decades ago, linguists established rules to identify the nature of code-mixing in the text [123, 81, 114, 84]. They also put forth rules we must follow when processing or generating the code-mixed dataset. These rules help to build better dialog systems on the code-mixed data. A few of the notable constraints are presented in the linguistic analysis.

- 1. Most speakers who use code-mixing for communication are equally fluent in both languages. Speaker believes that the mixed sentences are fairly acceptable to the listener.
- 2. During the interaction, the code-mixed utterances are spoken without hesitation.
- 3. Speakers and hearers usually know the dominant language from which the mixed utterance originated. The dominant language is known as Matrix Language, and the other language is known as Embedded language.
- 4. The native language of the group is generally the dominant language. Here the native language is Telugu, and the embedded language is English.
- 5. The medical terms used in the code-mixed conversations are mostly in English.

We analyzed the code-mixed nature of our corpus by comparing it with the constraints established by [114, 81, 123, 84].

- The paper [114] presented a Matrix language-frame model. The MLF model posits a Matrix and Embedded Language in a code-mixed sentence. In this case, elements of the embedded language are inserted into the morphosyntactic frame of the matrix language. A matrix language can be the speaker's first language or the language in which morphemes or words are more frequently used in speech. By analyzing the sentence's grammatical structure, we have observed that Telugu is the matrix language used in our utterances, and English is the embedded language.
- 2. The paper [81] presented a constraint that the root node of matrix language cannot be switched. The grammar of the code-mixed utterance must be the grammar obtained from the matrix language.
 - For example, in the code-mixed utterance where Telugu is a matrix language and English, is the embedded language, "fever *vacciMxi* (fever came)" is considered as the correct formation of code-mixed utterance. In contrast, the vice-versa "*jvaraM* came (fever came)" is considered incorrect. Furthermore, during data analysis and transcription, we observed that utterances such as "fever *vacciMxi*" had a bi-gram co-occurrence rate of 25%, whereas we did not observe any instances of "*jvaraM* came".
- 3. According to [81], closed class items (e.g., determiners, quantifiers, prepositions, possessives, aux, tense, helping verbs, etc.) cannot be switched.
 - For example in the utterance taken from the corpus: "*bAgA* stomach pain *vaswuMxi*" is considered as the correct code-mixed utterance as the quantifier "*bAgA* (heavy)" and tense "*vaswuMxi* (coming)" is present in the matrix language, whereas the vice versa is considered as an incorrect sentence. Most of the code-mixed sentences followed this pattern.
- 4. The paper[123], presented a constraint that intra-word switching occurs in code-mixed sentences. We have observed the instances that satisfy the constraint, such as red*gA*, yellow color*lo*, side*ki* where *-lo*, *-ki* are Telugu case-markers and *-gA* adverbial marker.
- 5. The paper [123] states that code-mixing occurs at a point where the mixing of two languages doesn't violate the surface syntactic structure of either languages. Examples are,
 - day by day *itla avuwuMxi*
 - malYlYi scan wIswe weVluswaxi kidney stones uMxo lexo ani.

The outcomes of the above analysis are,

- 1. According to [114], Matrix language-frame model suggested using the multilingual BERT model or the pre-trained Telugu and English fastText embeddings while processing the text.
- 2. The paper [123] on intra-word switching implied us to opt for sub-word tokenization or sub-word level embeddings. So that the morphologically altered words will be tokenized and we can extract more useful embeddings.



Figure 3.3 Intents and their distribution in the corpus

3.6 Annotation

In this section, we present the annotation of the dataset. The dataset is annotated with intents and slot tags. The annotation helps the NLU module interpret the utterance's intent and extract relevant slots from the utterance.

3.6.1 Intents

The notion of intent starts from the idea that the utterances can be assigned to a small set of Dialog Acts [153]. Recently, the intents have been adopted to identify a task or an action that the system can execute in a certain domain. Intents may cater to specific needs of the user (such as patient's complaints, doctor's responses) or to general needs (such as greeting, acknowledgments) [101]. We have used the DAMSL annotation schema [40] and annotation schema developed for medical dialogs [78, 133] to tag our utterances with their respective intent. We have chosen most of the DAMSL tags for our tagset as they are well suited for two-agent task-oriented dialogs, in which the participants collaborate to solve some problem. *Our intent tags record the utterance's purpose and its role in the dialog.* The intents, their definition, and distribution in the corpus are given in Table A.1 and in Figure 3.3. Table 3.7 gives an example of the intent annotated corpus.



Figure 3.4 Top 15 slots and their distribution in the corpus

3.6.2 Slots

In a doctor-patient conversation, we need to identify all the units of information that will help the dialog system solve a particular task, such as automated diagnosis or medical entity extraction. These relevant units or phrases of information are known as slots. Labeling the tokens with relevant slots is called slot annotation. Slot annotation will enable us to gather the information needed from an utterance.

Before slot annotation, we interacted with doctors to decide on the slot entities necessary to identify the relevant information in doctor-patient interaction. The slot list is then finalized by exploring the semantic information of our corpus's medical terminologies. We have also used WordNet's hypernym relation to understand the semantic information of the medical terminologies. Our slot entities are obtained from the medical terminologies from the top hierarchy introduced in MeSH(Medical Subject Headings)⁴ [57] and SNOMED CT (Systemized Nomenclature of Medicine - Clinical Terms)⁵ [42]. MeSH and SNOMED CT are the controlled medical vocabularies that store a collection of medical terms in human and veterinary medicine. They contain medical terms with synonyms and definitions covering anatomy, diseases, symptoms, procedures, microorganisms, substances, etc. These vocabu-

⁴https://en.wikipedia.org/wiki/Medical_Subject_Headings

⁵https://en.wikipedia.org/wiki/Systematized_Nomenclature_of_Medicine

	Utterance/word	Annotated tags
intent	d: xaggu xammu uMxA?	INFO_REQUEST
	(Is cough asthma present?)	
slot		
	xaggu (cough)	B-symptom
	xammu (asthma)	B-medical_condition
	<i>uMxA</i> ? (is present?)	0
intent	p: <i>xammu lexu</i> sir <i>xaggu</i> .	ANSWER
	(No asthma, only cough)	
slot		
	xammu (asthma)	B-medical_condition
	<i>lexu</i> (no)	B-negative
	sir	0
	xaggu. (cough.)	B-symptom
intent	p: xagge sir, xaggu naMju vaswuMxi xaggu wone	STATEMENT
	(Only cough is present sir, mucus is coming due to cough.)	
slot		
	<i>xagge</i> (only cough)	B-symptom
	sir,	0
	xaggu (cough)	B-symptom
	<i>naMju</i> (mucus)	B-body_substance
	vaswuMxi (coming)	B-symptom
	<i>xaggu</i> (cough)	B-symptom
	wone (due to)	I-symptom

Table 3.7 Intent and slot annotation in our dataset

laries are considered the standard medical vocabularies used to index medical journals and electronic health records for research purposes. In addition to the medical slot entities obtained from the medical vocabularies, we have included a few slot entities that provide more semantic information that can be used by a dialog system when parsing the sentence, such as the "time period" of the medical condition or symptom, "intensity" of suffering, etc. During slot annotation, we have also used positive or negative slot tags to indicate if the patient suffers from the symptom or not. A detailed list of slot entities with

their definition attributes and their distribution in the corpus is given in Table A.2. Top 15 slots are given in figure 3.4. After deciding the slot entities, we then followed the BIO (begin-in-out) schema [127] for slot annotation.

3.6.3 Annotation Analysis

We have created annotation guidelines with intent, slot names, and descriptions to facilitate proper annotation. Then the utterances were annotated by student research assistants in sequential order. The annotators assigned a DA tag to each utterance and slot tags for all the tokens in the utterance. At least two annotators annotate each dialog. The third annotator judges any further inconsistency.

We use Fleiss's kappa κ [59] to measure the inter-annotator agreement. For intent annotation, the utterance level agreement (κ) between the annotators is 0.79. For the slot annotation, the token level agreement between the annotators is 0.71, indicating a substantial agreement between the annotations and proving that our annotated data is reliable for further computation.

In the intent tagset, we can see that STATEMENT is the most frequent tag in the corpora, which agrees with the doctor-patient scenario, where the patients explain their medical condition to the doctor. The second-most frequent tag is INFO_REQUEST, where the doctor questions the patient to understand their medical condition or later seeks more information about their medical condition. Then comes the ANSWER utterance following the antecedent, i.e., INFO_REQUEST utterance. The doctor provides implicit acknowledgments (ACK) to notify their active participation in the conversation. The top 4 tags account for over 82% of the total corpus, suggesting imbalanced tag distribution.

During slot annotation, only explicit mentions of symptoms, body parts, disease duration, etc., are annotated as 'symptom', 'body_part', and 'time_period'. After annotation, we observed that the patient used explicit mentions of symptoms when the interaction started, followed by implicit mentions of symptoms. All of such implicit mentions are annotated as 'Outside' tags. The implicit conversations are relatively more than the beginning interactions, resulting in many "Outside" tags in our data. An example of intent and slot annotated sub_dialog is given in Table 3.7

3.6.4 Should we do Over-sampling and Under-sampling to handle the imbalanced data?

A dialog is an interaction in natural language between two or more agents. In a task-oriented dialog, the participants(doctor/patient - in our context) collaborate to solve a problem. In our task-oriented dialog, we see more questions (a.k.a INFO_REQUEST), answers, and acknowledgments compared to the other tags. It is because the participants want to progress towards a task by sharing information via questions, answers, and acknowledgments (a.k.a ACK). The same repeats for every dialog, thus making the dataset imbalanced. By nature, every task-oriented dialog data is imbalanced.

There are some notable approaches to balance the dataset. They are oversampling and undersampling.

- Oversampling involves increasing the instances of the minority class to balance the dataset w.r.t the majority class.
- Undersampling involves deleting some examples in the majority class to balance the dataset w.r.t the minority class.

The oversampling and undersampling are performed by various algorithms such as Imblearn's⁶ SMOTE(for oversampling) and undersampler(for undersampling) the data. There are multiple reasons why oversampling and undersampling should not be performed on the task-oriented dataset.

- If we use undersampling of the dataset, we should remove utterances related to INFO-REQUEST and ANSWER tags from the dataset. But these are the basis of information in our task-oriented dataset. By removing them, the dialog loses its purpose.
- If we perform oversampling, we create the copies of minority classes such as GREETINGS, AP-PRECIATE, SNU (signal not understood), SU (signal understood), CONV_CLOSE (conversation close, a.k.a end-of-conversation) on the above dataset. The disadvantage is that appending these instances to the data doesn't provide any information related to the task to the dataset. It may increase the likelihood of overfitting. It also increases the computational effort on the dataset.

Given the crucial medical domain, deleting the information via undersampling leads to losing critical medical information. Also, increasing the unnecessary information leads to overfitting the tags on the unnecessary information. Thus it leads to increased computational effort. Also, overfitting leads to a decrease in the classifier's performance.

Another question arises - Why don't we remove the tags of unnecessary information for better data balance? Removing low-frequency data tags won't affect the information present in the dataset for now. But when we encounter the utterances corresponding to these tags in a real-time dialog system. We may mistag the conversation, thus leading to a loss of information (present in tags such as RE-PEAT_REPHRASE, CORRECT), and it will also lead to a disturbance in the flow of conversation.

Thus, we do not use any oversampling and undersampling methods for data imbalance. We rely on advanced machine learning algorithms to handle the data. Without altering the data, we preserve the integrity and relevance of the dataset for the task-oriented dialog.

Also, there exist imbalances w.r.t each speciality in our dataset. Our dataset consists of more dialogs from the pediatrician, ophthalmologist, and otolaryngologist when compared to the other medical specialists. The dataset that we have collected depends upon the time period granted to us to collect the conversations and the number of patients visited during the time of data collection. But this did not impact the intent recognition and slot tagging on our dataset.

⁶https://imbalanced-learn.org/stable/index.html

3.7 Summary

In this chapter, we present the creation of our code-mixed dialog dataset 'Su-Vaid'. The dataset covers ten medical specializations, with more than 3k conversations and 29k utterances, and has a code-mixing index of 33%. The dataset is analyzed by various medical, conversational, and linguistic theories. Finally, The dataset is annotated with intent and slots for natural language understanding.

Chapter 4

Natural Language Understanding

This chapter talks about the natural language understanding module. NLU is the primary component of the dialog system. It usually encompasses two sub-tasks, namely intent classification (IC) and slot filling (SF). We present the baselines proposed for the both models. Later on we establish baselines using the SOTA NLU models and present an improvement over the baselines.

In ToD systems, natural language understanding (NLU) consists of identifying the main goals and information expressed in the user's utterances. It usually encompasses two sub-tasks, namely intent classification (also known as dialogue act classification) and slot filling (also known as slot labeling or slot tagging). Intent classification is a single-label or multi-label classification task where one or more intent labels must be assigned to the whole utterance of a user. The slot filling requires loading in predefined slots of information by extracting values from the content of the utterance.

4.1 NLU models

The NLU Baselines used can be broadly classified into 4 categories. Independent Models, Joint Models, and Contextual Models. Their description is given below.

4.1.1 Independent Models

Independent models treat IC and SF tasks separately. We have applied various RNN, CNN, and BERT architectures for IC and SF tasks on our dataset. As our data is multilingual, we have used multilingual pre-trained fastText embeddings [64], and BERT multilingual pre-trained embeddings [46]. The pre-trained embeddings capture a better semantic representation of the word when compared to the embeddings directly learned on the corpus [64, 46]. We have also learned the embeddings directly from our corpus using the Roberta Language model. Further details are given in section 4.1.1.

• **RNN**: The bi-directional Jordan RNN [80] is used for the IC and SF tasks. The representation of the last hidden layer is used for IC. Whereas the encoded information obtained from all the hidden units in the last layer is used for SF. [111].

- **CNN**: A sequence learning-based convolutional neural network (CNN) is used to extract features [61]. Here, each filter will slide across the sequence, from beginning to end, looking at all three consecutive tokens at a time. The embeddings are passed through convolutional blocks with a special activation function called Gated Linear Units (GLU). For the SF task, the convolved representation is given to a feed-forward unit to obtain the slot tag for each token in the sequence. Finally, the Max-pooling is applied to get the word representation before passing it to the output layer for the IC task.
- **Code-Mixed BERT**: We have trained¹ the language model from scratch using Roberta Transformer and BPE Tokenizer on our dataset. The BERT model is obtained from the transformers library ². Further, we fine-tuned the model for the SF and IC tasks.
- mBERT: It is a bidirectional transformer encoder model with pre-trained multilingual word embeddings [46]. It is a transformer-based self multi-headed attention model pre-trained on a huge data collection. This transfer learning is very successful when we learn a classifier from a dataset using pre-trained embeddings. We used the smaller version of the pre-trained multilingual BERT model called *bert-base-multilingual-cased* obtained from the transformers library. The accepted pre-trained model is fine-tuned for our intent and slot-tagged dataset.
- **IndicBERT**: A multilingual ALBERT model trained on a large-scale corpus, covering 12 Indian languages [85]. The model is fine-tuned on our dataset for IC and SF tasks.
- Clinical-BERT: Clinical-BERT is a *BERT*_{base} model trained on clinical notes to obtain embeddings specific to medical data [6]. The model is fine-tuned on our dataset for IC and SF tasks.

4.1.2 Joint Models

The NLU model needs both IC and SF tasks to capture the semantic frames in an utterance. Thus, it is intuitive to think that these tasks are correlated and benefit each other when trained jointly. Most joint modeling approaches use parameter sharing to capture the relationship between SF and IC. Using a joint model for the two NLU tasks simplifies the dialog system, as only one model needs to be trained and deployed.

• Joint RNN model: [100] use a neural sequence to sequence (encoder-decoder) model with an attention mechanism. The shared encoder is a Bi-LSTM (Bi-directional Long Short Term Memory), and the decoder then uses the last hidden state of the encoder to generate a sequence of slot labels. At the same time, there is a separate decoder for the IC. The attention mechanism is used to learn alignments between the slot labels in the decoder and the words in the encoder. We have used fastText pre-trained embeddings.

¹https://huggingface.co/blog/how-to-train

²https://huggingface.co/transformers/

- Slot Gated RNN: A slot-gated mechanism is introduced to capture explicit relations between the intent and the slots [63]. The Bi-LSTM-based attention model [100] is used along with the slot gate. The slot-gated model allows the slot filling to be conditioned on the learned intent result to achieve better joint slot filling and intent detection.
- Joint BERT: [33] used joint modeling with BERT as the underlying architecture for the joint modeling of SF and IC tasks. The encoded information obtained at [*CLS*] (a.k.a classification token) is given to the feed-forward network for intent detection. The encoded representations obtained from the other tokens of the utterances are given to the feed-forward network for slot tagging.
- **SlotRefine** [172]: The transformer architecture is used to construct a two-pass iteration mechanism to handle the problem of the uncoordinated slots caused by conditional independence of the non-autoregressive model.

4.1.3 Contextual models

A multiturn dialog exhibits a flow of conversation. Such sequential flow exhibits long-term dependencies in utterances. The manipulation of such flow helps the model capture long-term dependencies and improves the performance of NLU. Such models that encode previous knowledge are generally known as Contextual NLUs.

- CASA-NLU [67]: It is a context-aware self-attentive NLU (CASA-NLU) model. In addition to the current user utterance, the model uses previous utterances with their dialog acts, intents, and slot tags as contextual signals over a variable context window. The model also uses a directional self-attention-based encoder (DISAN) [144]. DISAN adds absolute position embedding of the previous and current utterances. The model obtains a one-hot representation of intent and slot ground truth labels of the previous utterances. The intent and slot one hot embedding are concatenated with their corresponding previous utterances. The attention mechanism is used to extract relevant components of the previous utterances. It is known as a contextual vector. The contextual vector and the current utterance are used for predicting the intent and the slot labels for the current utterance.
- **BERTNLU-Context**³ [185]: In this model, the last three utterances are given to the BERT model to incorporate the contextual information. The discrete set of previous utterances is separated with [*SEP*] tokens, and a [*CLS*] token is inserted initially. Then, it is passed through the MLP(Multi-Layer Perceptron) to predict the intent and the slot tags, respectively. We have considered two contextual models,

³This model was initially tuned on the chinese dataset thus we cannot use it. So we have used mBERT- i.e., *bert-base-multilingual-cased* as the underlying BERT model, instead of the BERT model suggested in the paper [185].

- 1. Memory(I) considers IC and SF tasks to be Independent.
- 2. Memory(J) considers the correlation between IC and SF tasks and uses the Joint modeling method.

4.2 Experimental Settings

The experiments and evaluations were run on NVIDIA GeForce RTX 2010 with GPU driver version 457.51, and CUDA version: 10.1 with 8 GB GPU memory. We used PyTorch and torch-text libraries to implement the RNN and CNN models. The training, validation, and testing data split was 75%, 6.25%, and 18.75%.

We used the default parameter settings mentioned in their original papers for the baseline models. We used 300-dimensional FastText and 768-dimensional BERT word embeddings for baseline models using pre-trained word embeddings.

The hyperparameters used to fine-tune the huggingface transformer models are, maximum sentence length = 232, batch size = 32, learning rate = 2e-5, epsilon = 1e-8, number of epochs = 10 and dropout = 0.1.

4.3 Performance of Multilingual Models

The performance of the baseline models on our dataset is given in Table 4.1.

For the independent models, given the intent classification task, we observed that the RNN model misclassified the STATEMENT and ANSWER, ANSWER, and ACK (ACKNOWLEDGEMENT) labels. One perception is that all these utterances can be visualized as statements that the patient or doctor utters during the conversation. The SOTA transformer architecture and pre-trained embeddings of the BERT model gave a better representation of our data, benefiting the slot-filling task.

The joint models have seen a decline in performance. It is because the intent tags are domainindependent, and the slot tags are domain-dependent. This inconsistency of domain dependence and independence affected the performance of the model. The slot-gate NN model [63] gave better results on the dataset. In the slot gate model, the value of the gate determines the flow of intent information to the slot. We have observed that the gate value is very low, signifying a low correlation between the slot and the intent for a given utterance. For further analysis, we have also appended the slot gate to the joint mBERT model and observed that the gate value is very low, thus proving a low correlation between the intent and the slot models.

Using contextual information while fine-tuning the BERT model improved the model's performance. The encoded knowledge obtained from the previous utterances helped the contextual model distinguish between the STATEMENT, ANSWER, and ACK tags, thus improving the model's performance.

The Memory(I) model performed better for the IC and SF tasks when compared to the Joint memory model. As mentioned above, this is due to the poor correlation between IC and SF tasks.

			IC					SF		
	Models	Р	R	F1	Acc	Р	R	F1	Acc	
Ι	Jordan RNN + FastText Embed	0.35	0.37	0.33	0.67	0.29	0.29	0.29	0.76	
	Jordan RNN + BERT Embed	0.36	0.37	0.34	0.68	0.29	0.30	0.29	0.77	
	CNN + FastText Embed	0.24	0.19	0.21	0.47	0.44	0.33	0.37	0.82	
	CNN + BERT Embed	0.24	0.19	0.21	0.48	0.44	0.33	0.37	0.81	
	Code-Mixed BERT	0.38	0.42	0.40	0.72	0.44	0.48	0.46	0.85	
	mBERT	0.42	0.46	0.44	0.73	0.52	0.58	0.54	0.94	
	IndicBERT	0.33	0.37	0.34	0.70	0.31	0.37	0.31	0.86	
	clinical-BERT	0.31	0.36	0.33	0.67	0.22	0.31	0.23	0.80	
J	Joint RNN model + FastText Embed	0.30	0.37	0.33	0.67	0.27	0.31	0.29	0.74	
	Joint RNN model + BERT Embed	0.30	0.37	0.33	0.67	0.27	0.31	0.29	0.75	
	Slot Gated RNN + FastText Embed	0.38	0.41	0.39	0.72	0.43	0.49	0.46	0.84	
	Slot Gated RNN + BERT Embed	0.38	0.41	0.39	0.72	0.43	0.49	0.46	0.84	
	Slot Refine + FastText Embed	0.36	0.40	0.38	0.70	0.32	0.38	0.36	0.75	
	Slot Refine + BERT Embed	0.36	0.40	0.38	0.71	0.31	0.39	0.36	0.75	
	Joint BERT	0.36	0.41	0.38	0.69	0.39	0.40	0.38	0.72	
M(I)	BERTNLU-context	0.54	0.58	0.56	0.87	0.52	0.58	0.54	0.95	
M(J)	CASA-NLU	0.37	0.42	0.39	0.71	0.39	0.41	0.40	0.72	
	BERTNLU-context	0.39	0.43	0.41	0.75	0.39	0.41	0.40	0.73	

Table 4.1 Performance of Baseline models on our dataset, Where I refers to Independent Model, J refers to Joint Model, M refers to Memory Model, IC refers to Intent Classification, SF refers to slot Filling and P, R, F1, A stands for macro-Precision, macro-Recall and macro-F1 measure, and accuracy

The slot tagging showed no improvement in performance when previous utterances were used. It denotes that only current utterance is essential for slot filling on our dataset.

In medical documents, false negatives should be avoided since they can affect the patient's diagnosis. Thus, recall value should be more than precision when analyzing medical data. Our models also exhibited a similar behavior by delivering better recall values when compared to the precision. It suggests that our fine-tuned models minimized the false negatives in the dataset.

4.4 Cross-lingual XLMR Models

		IC				SF			
	Models	Р	R	F1	Acc	Р	R	F1	Acc
Ι	Jordan RNN + MUSE Embed	0.35	0.37	0.33	0.67	0.29	0.29	0.29	0.76
	Jordan RNN + XLM-R Embed	0.35	0.37	0.33	0.67	0.29	0.29	0.29	0.75
	CNN + MUSE Embed	0.24	0.19	0.21	0.47	0.35	0.33	0.34	0.80
	CNN + XLM-R Embed	0.24	0.19	0.21	0.48	0.35	0.33	0.34	0.80
	XLM-R	0.42	0.46	0.44	0.73	0.52	0.57	0.53	0.93
J	Joint RNN model + MUSE Embed	0.30	0.37	0.33	0.65	0.27	0.31	0.29	0.73
	Joint RNN model + XLM-R Embed	0.30	0.37	0.33	0.67	0.27	0.31	0.29	0.75
	Slot Gated RNN + MUSE Embed	0.36	0.38	0.37	0.70	0.43	0.49	0.46	0.83
	Slot Gated RNN + XLM-R Embed	0.36	0.38	0.37	0.71	0.43	0.49	0.46	0.84
	Slot Refine + MUSE Embed	0.36	0.40	0.38	0.69	0.32	0.38	0.35	0.72
	Slot Refine + XLM-R Embed	0.36	0.40	0.38	0.70	0.32	0.38	0.35	0.72
	Joint XLM-R	0.36	0.40	0.38	0.69	0.32	0.38	0.35	0.72
M(I)	XLM-R-NLU-context	0.54	0.58	0.54	0.84	0.52	0.55	0.53	0.93
M(J)	XLM-R-NLU-context	0.36	0.41	0.38	0.69	0.35	0.40	0.37	0.73

Table 4.2 Performance of Cross-lingual Baseline models on our dataset. Where I stands for Independent Models, J stands for Joint Models, and M stands for Memory Models. P, R, F1, A stands for macro-Precision, macro-Recall, and macro-F1 measure, and accuracy.

As mentioned above, we have utterances with bilingual words in our data. Logically, such words should have the same embeddings when dealing with cross-lingual datasets. Cross-lingual word embeddings (CLWEs) can be used to align the embeddings of bilingual and tri-lingual words. CLWEs are the same as monolingual word embeddings, but it captures words from multiple languages into one embedding space. CLWEs use monolingual corpora, dictionaries, and parallel corpora to do the projection or alignment [134].

We used MUSE, a python library for multilingual unsupervised and supervised embeddings [39, 95] to obtain the cross-lingual word embeddings. We opted for a supervised model that uses seed

dictionaries to learn cross-lingual mapping. The cross-lingual mappings are learned between Telugu and English languages with Telugu-English⁴ seed dictionaries.

We have also used XLM-R, a multilingual transformer model that achieved strong performance on cross-lingual understanding tasks. It relies on the Masked Language Model's objective and can process text from 100 separate languages. The XLMR model is trained on 2.5TB of cleaned CommonCrawl data. XLM-Roberta uses the one large shared Sentence Piece model. XLM-R's training routine is the same as the monolingual Roberta model. We have obtained the pre-trained XLM-R model known as "xlm-roberta-base"⁵ from the transformers library and used it to fine-tune our dataset.

Table 4.2 shows our dataset's performance of cross-lingual baselines.

4.5 Performance of Cross-lingual NN models

The models with cross-lingual embeddings have shown no improvement in performance when compared to the baseline monolingual models.

Even the XLM-R model has shown no improvement in accuracy compared to the mBERT model. Similar to the BERT model, XLM-R misclassified the STATEMENT and ANSWER labels.

We have seen no improvement in performance when cross-lingual embeddings are used on the joint models because of the independence between the intent and slot tags.

When cross-lingual embeddings are used, the memory models have shown a decrease in classification accuracy for DA tagging and slot tagging tasks.

The XLMR models did not perform better when compared to the mBERT model on our dataset. The results prove that further research and more training data are necessary to capture the cross-lingual information in multilingual and code-mixed datasets.

4.6 Contextual Signals and Chunk Identification for NLU

We improved the performance of the traditional mBERT model by augmenting it with contextual signals for intent classification and by projecting the slot-filling task as a chunk identification problem.

4.6.1 mBERT with Contextual Signal for IC

Figure 4.1 presents the architecture of the IC model on the dataset. For the IC task, the model takes a discrete set of previous utterances with their ground truth labels and the current utterance and encodes them using an mBERT model. The utterances are ordered as current to previous k utterances, and they are separated with [SEP] token. We insert a [CLS] token at the beginning for intent classification. The

⁴https://te.wikipedia.org/wiki

⁵https://huggingface.co/transformers/model_doc/xlmroberta.html



Figure 4.1 mBERT with contextual signal for Intent Classification

encoded information of the BERT model is given to the multi-layer perceptron (MLP) to capture the intent.

In the test phase, each predicted previous intent tag in a dialog is stored and given as a contextual signal while predicting the intent of the current utterance.

4.6.2 Chunk Identification - SF Model

[172] argues that identifying the slot chunks and dependencies among slot chunks is enough. We used the same approach to identify token dependencies in slot chunks. The architecture of the slot tagging model is given in Figure 4.2.

We extend the mBERT model to construct the two-pass mechanism. The mBERT and vanilla transformer models extract the coarse-grained slot tags (without BIO-tag information) in the first pass. In the second pass, the predicted slot tags are passed along with the token embedding of the utterance to predict the fine-grained slot tags (i.e., slot tags with BIO-tag information). In detail,

1. In the first pass,

(a) The initial slot tags are set to (Outside) O-tag.



Figure 4.2 Bourndary aware Slot Filling Model using 2 phase mechanism. tp, s, and pos refer to time_period, statement, and positive tags, respectively.

- (b) The O-tag and token embeddings are given to the mBERT model and the vanilla transformer model to predict the coarse-grained slot tags.
- 2. In the second pass,
 - (a) The coarse-grained slot tag and the token embeddings are given to the model.
 - (b) The model has to predict the slot-tag boundaries, i.e., the BIO tags for the given slot tags.

This two-pass mechanism forces the model to learn the fine-grained slot tags (i.e., slot tags with BIO information). Thus, the predicted slot labels become more consistent with identifying the chunks.

These two iterations share the model and require no extra parameters. Intuitively, in doing so, the model initially generates a draft of coarse-grained slot tags in the first pass. In the second pass, the model is forced to learn the consistent BIO tags by adding coarse-grained slot tag information along with the tokens in an utterance. Thus, the predicted slot labels become more consistent with the help of boundaries. Thus, explicitly modeling the coarse-grained slot tags in the first pass and fine-grained slot tags in the second pass helps the model tag the qualitative information related to the slots and simplifies the slot tagging in the next pass.

4.6.3 Evaluation of our approach

	Table 4.3 Performance of our IC and SF models on the dataset									
Model macro-P macro-R macro-F1										
IC	mBERT with contextual signal	0.63	0.68	0.66	0.96					
SF	Chunk Identification - SF Model	0.52	0.58	0.55	0.97					



Figure 4.3 Comparision between BERTNLU-context and mBERT with the contextual signal on the dataset

Table 4.3 shows the performance of mBERT with the contextual signal on our dataset. During the test phase, the batch size is kept as "1", and the utterances are modeled sequentially so that the previous utterance's intent is considered for predicting the intent of the current utterance. The contextual window considered is 3 (This value is considered after observing the conversational flow during the annotation process). We observed that adding intent ground truth labels during training and considering previous



Figure 4.4 Confusion matrix - BERTNLU-context for intent tagging.

intent during the test phase helped the utterance establish a well-determined context and improved its classification accuracy.

The performance of BERTNLU-context and the mBERT with the contextual signal is shown in Figure 4.3. As expected, contextual signals improved IC performance for most labels compared to the BERTNLU-Context. The improvement in accuracy for most of the labels shows the impact of contextual signals on the multi-turn conversational dataset.



Figure 4.5 Confusion matrix - mBERT with contextual signal for intent tagging.

Figures 4.4 and 4.5 present the confusion matrices of BERTNLU-context and mBERT with the contextual signal on the given dataset. The BERTNLU-context model misclassified ANSWER and ACK as STATEMENT labels on our dataset, whereas the enhanced approach performed better. It proves that ground truth intent labels improve the intent prediction on our dataset.

The performance of naive mBERT and chunk identification SF models for each label is given in Figure 4.6, and for each epoch is given in Figure 4.7. As the chunk carries most of the slot-related information in the dataset, balancing the model with a 2 phase mechanism reduced the number of incorrectly



Figure 4.6 Performance of naive mBERT and chunk identification SF models on the dataset for each slot label (Only a few slots were used for clear observation)

classified tags. We can see the improvement in most intermediate(I) chunks because of the model's 2 phase chunk identification mechanism. It improved the performance of the dataset by 3%. Thus proving that our model achieves better chunk identification when compared to the naive mBERT model.

4.6.4 Adapter-Transformers

The transformer model achieves better results for intent and slot tagging, But the problem with this deep learning model is it involves downloading and fine-tuning pre-trained models consisting of hundreds of parameters. Training such large pre-trained models is expensive, slow, and time-consuming on the dataset. [119] developed Adapters, a.k.a small bottleneck layers, inserted within each layer of a pre-trained model to alleviate this issue by avoiding full fine-tuning of the entire model. The adapter framework uses the steps mentioned below to introduce adapters in the transformers.

- 1. Adding new adapter configuration and weights to the pre-trained transformer.
- 2. Freezing the pre-trained weights and only fine-tuning the adapter weights for a specific task.
- 3. Loading and storing the adapter weights for the future.



Figure 4.7 Overall accuracy performance per epoch for naive mBERT and chunk identification SF models on the dataset

They also proposed a framework that allows a dynamic mix-in of pre-trained adapters for different tasks and languages known as AdapterHub [120]. The framework built on top of the popular Hugging-Face Transformers library enables extremely easy and quick adaptations of state-of-the-art pre-trained models (e.g., BERT, RoBERTa, XLM-R) across tasks and languages.

Experimental Settings - Adapters

We have implemented the intent classification and slot tagging models by considering the adapter transformers. We have used the adapter-transformers library⁶ that is built over the hugging-face transformers library for testing the adapter models. We used the BertAdapter model, which contains adapter modules for the BERT models. We used the "bert-base-multilingual-cased" pre-trained checkpoint from the hugging-face transformers library. We added classification or tagging head to the adapter model for intent and slot tagging. We use the default 'Pfeiffer' configuration [119] in the adapter models.

Analysis of Adapters

The pre-trained models used by our dataset consisted of 117 million parameters, and our model took *30 minutes to complete a single iteration* on the dataset. Only 0.2 million parameters are added to the

⁶https://docs.adapterhub.ml/index.html
				IC				SF		
	Models		Р	R	F1	Acc	Р	R	F1	Acc
Ι	mBERT with adapters		0.34	0.33	0.33	0.71	0.52	0.56	0.52	0.92
	XLM-R with adapters		0.32	0.33	0.32	0.70	0.52	0.56	0.52	0.91
J	Joint BERT with adapters		0.31	0.35	0.32	0.70	0.39	0.40	0.38	0.72
	Joint XLM-R with adapters		0.31	0.34	0.31	0.69	0.32	0.38	0.35	0.72
M(I)	BERTNLU-context wit	h	0.38	0.36	0.37	0.74	0.52	0.53	0.52	0.92
	adapters									
	XLM-R-NLU-context with	h	0.38	0.36	0.36	0.72	0.51	0.53	0.51	0.92
	adapters									
M(J)	BERTNLU-context wit	h	0.28	0.32	0.29	0.73	0.39	0.41	0.40	0.73
	adapters									
	XLM-R-NLU-context wit	h	0.28	0.31	0.28	0.71	0.39	0.41	0.40	0.73
	adapters									
*	mBERT with contextual sign	al	0.31	0.39	0.32	0.74	0.52	0.58	0.52	0.92
	for IC and Chunk identification	n								
	for SF with adapters									

Table 4.4 performance of adapter-transformers on our dataset. Where I stands for Independent Models, J stands for Joint Models, M stands for Memory Models, * refers to our approach, and P, R, F1, and Acc refer to macro-Precision, macro-Recall, and macro-F1 score and Accuracy score, respectively.

model when adapter models are used. As the pre-trained transformer weights were frozen during finetuning, we saw a huge decrease in the time taken for each epoch. The model took *less than 15 minutes for each epoch* for the intent classification task and *less than 10 minutes for each epoch* for the slotfilling task. I.e., the time taken is decreased by more than half for each epoch using adapters. However, we saw a decrease in the classification metrics, as shown in Table 4.4.

Our code-mixed data is varied at the morphological level. The transformer models are trained on monolingual tasks. It would be difficult for the pre-trained transformer models to capture the semantic information of the code-mixed words. Thus the pre-trained model needs to be fine-tuned on the given codemixed dataset. But as the adapters freeze the learning of pre-trained models, the essential information captured by attention heads while fine-tuning the models is missing. It led to a decreased performance in the classification metrics. Also, the model should be highly reliable in critical domains

like medicine. Therefore, given the code-mixing in the medical domain, we should fine-tune the models on the relevant training data before deploying them.

4.7 Summary

In this chapter, we briefly described the NLU and its subtasks. We presented SOTA NLU baselines on our dataset. As our data is code-mixed, we considered both multilingual and cross-lingual models for establishing baselines on our dataset. We also improved the performance of the NLU models by adding contextual signals for intent classification, and we presented the slot-filling task as a chunk identification problem.

Chapter 5

Hate Speech Detection

This chapter presents our work on hate speech detection on code-mixed datasets.

Hate speech is defined as any communication that disparages a person or a group on the basis of some characteristic such as race, color, ethnicity, gender, sexual orientation, nationality, and religion [140].

Healthcare workers are at greater risk of workplace abuse than most other workers, with nurses and family physicians rated as most at risk of abusive encounters with patients [45]. Statistics show that over 75% of doctors across the country have faced at least some form of violence and that 68.33% of the violence was committed by the patient's attenders /escorts. The doctors have witnessed hate speech when conversing with the patients too. This scenario was observed when the patients insulted other doctors due to the delay in treatment or if the patient was given incorrect treatment.

A few examples of patients using hate speech towards doctors are presented in Table 5.1.

With the increase in social media access, Offensive content, and hateful material on the internet has increased in the recent past. The internet harbors a variety of hateful and offensive statements, and nowadays, social media is a hotbed of such conversations. Recently, countries across the world have already begun to address hate speech and offensive content and how it affects society's functioning. Research and technologies worldwide are utilizing natural language and machine learning tools to detect and curb the use of offensive content on social media [29].

Automatic detection of Hate speech is crucial in conversations. Modern dialog systems are often designed to process and train the models even during the conversation simulation. If the hate speech is not addressed in the dialog system, the dialog system will process the hate speech conversations and simulate them and respond to the users offensively. This scenario was observed in the TAY bot released by Microsoft. Tay was an artificial intelligence chatbot that Microsoft Corporation released via Twitter on March 23, 2016; it caused subsequent controversy when the bot began to post inflammatory and offensive tweets through its Twitter account, causing Microsoft to shut down the service only 16 hours after its launch.

Our data collection found no data related to hate speech in doctor-patient interactions. So we trained the hate speech detection module by considering the current code-mixed datasets. We also evaluate the

Utterances

ఏ కులపొడివి అయ్యా నువ్వు? Imply: which caste do you belong to. Comment: caste discrimination నువ్వు చూడడానికే పొట్టిగా, నల్లగా ఉన్నావు. ఇక నన్ను ఏం చూస్తావు? You look short and dark. How will you treat me? Comment: Body shaming. ఇంట్లో కూర్చోవాలే కానీ, doctorలు అయ్యి hospital కు ఎందుకు వస్తారమ్మ? You should sit at home. Why do you (women doctors) come to hospital? Comment: gender discrimination



performance of the hate speech detection modules on those datasets. These datasets are available in Telugu, Tamil, Malayalam, Kannada, and Hindi code-mixed languages. The model that is fine-tuned on the Telugu corpus [137] is used in our dialog system.

5.1 Proposed Models for Code-Mixed Hate Speech Detection

In this section, we present our proposed models for hate speech detection.

5.1.1 Transformers with the Class Balanced Loss for Offensive Language Identification in Dravidian Code-Mixed text.

In the paper [53], we used multilingual pre-trained BERT with the class-balanced loss for codemixed hate speech detection. The Class-Balanced Loss assigns weights to the loss based on the effective number of samples in the given data [44]. We used a re-weighting scheme that uses sufficient samples for each class to re-balance the loss, thereby yielding a class-balanced loss. We also pre-process the code-mixed dataset to extract reliable information. We begin the model by primarily pre-processing the data.

Pre-processing

The given code-mixed dataset depicts real-time scenarios of variations in the spelling, script changes, and use of hashtags, mentions, and emoticons in the text and has imbalance problems. So pre-processing is necessary for such a dataset. During pre-processing,

- To resolve the ambiguities resulting from the script change, we back-transliterated the script to the native language. As the data has Dravidian and English comments, we used the NLTK¹ English word corpus to detect if the word is in English or not. Later we back-transliterated the word to its native script. We applied linguistic rules not to transliterate the tweet/comment "not-Kannada/not-Tamil/not-Malayalam" as they are not in the intended language.
- We removed all the punctuations, URLs, mentions, unwanted numbers, and emoticons from the given dataset. We accepted repetitions of characters up to a length of 2 and removed others.

Class balanced loss to handle dataset imbalance

While handling an imbalanced dataset (one with most of the samples belonging to very few classes and many other classes with very few instances), loss calculation can be tricky. The most common approach to balancing the loss is assigning weights to the loss. The weights are calculated as the inverse of the number of class instances or the inverse of the square root of the number of class instances. This weighing scheme creates the problem by shifting focus entirely to the classes with very few instances.

To handle the shift, the authors [44] proposed a Class-Balanced Loss based on the Effective Number of Samples. The effective number of samples is defined as the volume of samples. It can be calculated by a simple formula $(1-b^n)/(1-b)$, where n is the number of samples and b is a hyper-parameter, and it takes values between [0,1]. The authors designed a re-weighting scheme that uses sufficient samples for each class to re-balance the loss, thereby yielding a class-balanced loss.

$$CB(\mathbf{p}, y) = \frac{1}{E_{n_y}} \mathcal{L}(\mathbf{p}, y) = \frac{1 - \beta}{1 - \beta^{n_y}} \mathcal{L}(\mathbf{p}, y)$$
(5.1)

Here, $\mathcal{L}(\mathbf{p}, y)$ can be any loss function. n_y is the number of estimated samples for each labels y.

5.1.2 Using BERT and multilingual BERT models for hate speech detection (with data augmentation)

We [52] used pre-trained bi-directional encoder representations using transformers (BERT) and multilingualBERT for hate speech and offensive content detection for English, German, and Hindi languages. We leveraged the datasets from the shared task [106] while fine-tuning our classifier. Such data-augmented learning helped BERT learn better representations, thereby improving the model's performance.

¹https://www.nltk.org/

5.1.3 Hate Speech Detection on Code-Mixed Dataset Using a Fusion of Custom and Pretrained Models with Profanity Vector Augmentation

We [56] implemented a fusion model that takes advantage of a code-mixed model trained explicitly on the code-mixed dataset and a pre-trained multilingual model. This fusion helps the model learn a better semantic representation of the code-mixed data. Our method also includes a profanity word list collection and profanity vector augmentation. Intuitively, a custom code-mixed model provides a fine-tuned supervision of downstream tasks, while multilingual models are employed for knowledge transfer across languages. The profanity vector helps improve the language modeling of the data by emphasizing the profane words used in each comment. Along with model training and fine-tuning, we initially pre-process the code-mixed data to deal with variations in spelling and transliteration.

A detailed description is presented below.

Pre-processing

This section presents the steps performed for transliterating the code switched data to its respective languages.

The given code-mixed dataset depicts the real-time scenario of variations in spelling, transliterations, script changes, emoticons, URLs, hashtags, and mentions. So pre-processing is necessary for such a dataset. During pre-processing,

- 1. To resolve the transliteration variations resulting from the script change, we back-transliterated the script to the native language. We have analyzed that the code-mixed data has the matrix and embedded languages, where the embedded language is mostly English. Firstly, we used the NLTK² English word corpus to detect if the word is in English or not. If the word is not from the English language, we assume it's a foreign word/native word. We used google trans API³ to detect the word's language id. Later, we back-transliterated the non-English word to its native script⁴. Of course, this won't correctly capture the correct native word because code-mixed languages have a lot of variations in spelling. But as we use sub-word tokenization, we expect the BERT model to capture accurate semantic information. We also applied linguistic rules not to transliterate the tweet/comment "not-Kannada/not-Tamil/not-Malayalam" as they are not in the intended language.
- 2. To resolve spelling variations in the roman words, we used the google trans API to detect the language of the word. We used symspellpy⁵, a python library, to correct the spelling mistakes in the English corpus.

transliterates the given text to the one specified by the source/target language.

²https://www.nltk.org/

³https://pypi.org/project/googletrans/

⁴If the source and target languages are kept the same in the API, the google trans API works as a transliterator. i.e., it

⁵https://pypi.org/project/symspellpy/



Figure 5.2 Training the code-mixed model from scratch on the code-mixed dataset

3. Using regex codes, we removed all the URLs, mentions, hashtags, emoticons, unnecessary punctuations, unwanted numbers, and emoticons from the dataset. In addition, we accepted repetitions of characters up to a length of 2 and removed others.

Custom Code-mixed model: Training a Model From Scratch

The BERT models are trained on the monolingual corpus. They won't cater to the needs of codemixed data. So we trained a Roberta model from scratch by using code-mixed training data. We used a byte-level Byte-pair encoding tokenizer to tokenize the comments or posts. The tokenizer extracted vocabulary from the code-mixed data. Then we trained a Roberta language model on the code-mixed data. Figure 5.2 gives the training of the code-mixed model from scratch. The process of tokenizing and training the language model using the hugging face-transformers library is available here⁶.

Multilingual BERT

Multilingual BERT [46] is a transformer model [157] that is trained on Wikipedia text across 104 languages. It uses a non-regressive model. It analyzes the meaning of a term depending on its context. As the model is pre-trained on a large corpus, the semantic and syntactic information is well modeled, and it can be directly fine-tuned for a specific task. The transformer part in the BERT works like an attention mechanism capable of learning the contextual relationships between the terms in a sentence. The BERT uses the [CLS] token, also known as the classification token, to indicate the beginning of the sentence. The last hidden state of this token is used to classify the given sentence. Embeddings from

⁶https://huggingface.co/blog/how-to-train

 $BERT_{base}$ have 768 hidden units. The BERT configuration model takes a sequence of words/tokens at a maximum length of 512, producing an encoded representation of dimensionality 768.

Profanity Vector

The profanity vector was first introduced in the hate speech detection model proposed by [36]. The model considers the profanity dataset obtained from the paper [108]. But it is not scalable to all the datasets as the code-mixed hateful words are dependent only on [108]. So we have proposed a new method of finding the profanity words and augmenting the profanity vector to the given model.

- 1. Finding the profanity words
 - (a) Initially, we took a list of unique words for transliterated datasets.
 - (b) We translated the non-English words to the English language using googletrans API.
 - (c) We know that hateful words have a negative polarity. We have used sentiment intensity. analyzer⁷ from the NLTK library to get the polarity of each word.
 - (d) We also take the list of hateful words from the CMU Offensive/Profane Word List⁸.
 - (e) We identify the polarity of each unique word using the sentiment polarity analyzer, and we also verify if the word is present in the hate database. If the word is of negative polarity or the word is present in the hate database, we consider that word as the hate word.
 - (f) For each dataset, we maintain a profanity words list
- 2. The profanity word vector.
 - (a) The profanity word list is used to construct the profanity word vector.
 - (b) We used the same approach as given in [36] to construct the profanity vector. For each tweet, we add a corresponding 1 that indicates the presence of a hate word. In contrast, its absence is indicated by 0 to emphasize the absence of a contextually subjective profane word.

Method

In this section, we fine-tune the code-mixed and pre-trained BERT models using the hate speech and offensive language training datasets. The proposed method is shown in figure 5.3

We consider the code-mixed training data and its labels to fine-tune a code-mixed classification model from the given language model. The code-mixed model learns the task-specific knowledge since it is trained and fine-tuned on code-mixed data. Further, the pre-trained BERT model is fine-tuned on the code-mixed data. The pre-trained models are effective on unseen test data as they are trained on

⁷https://www.nltk.org/api/nltk.sentiment.html

⁸https://www.cs.cmu.edu/~biglou/resources/bad-words.txt



Figure 5.3 The proposed method

Dataset	No. of comments	Labels
Dravidian Languages-EACL	20,010	Not Offensive, Offensive Untargeted, Offen-
2021 Malayalam-English		sive Targeted Individual, Offensive Targeted
dataset [27]		Group, Offensive Targeted Other, Not Malay-
		alam.
HASOC-DravidianCodeMix	5,000	Not Offensive, Offensive
Malayalam-English dataset [31]		
Dravidian Languages-EACL	43919	Not Offensive, Offensive Untargeted, Offen-
2021 Tamil-English dataset[27]		sive Targeted Individual, Offensive Targeted
		Group, Offensive Targeted Other, Not Tamil.
HASOC-DravidianCodeMix	4,000	Not Offensive, Offensive
Tamil-English dataset [31]		
Dravidian Languages-EACL	7772	Not Offensive, Offensive Untargeted, Offen-
2021 Kannada-English dataset		sive Targeted Individual, Offensive Targeted
[71]		Group, Offensive Targeted Other, Not Kan-
		nada.
Bohra Hindi-English Code-	4575	no, yes
Mixed dataset [16]		
HEOT (Hindi-English Offensive	3679	nonoffensive, abusive, hate-speech
Tweet) dataset [108]		

 Table 5.1 Dataset Statistics

large corpora. The concatenation of the last hidden states of the code-mixed classification model, pretrained BERT model, predicted probabilities and profanity vector is provided to fine-tune a feed-forward network. This feed-forward network learns the attention-specific weights given to the models. Then the model is fine-tuned further to predict the correct labels.

5.2 Datasets

This section lists available hate speech detection datasets where each comment or post is assigned to a label.

Malayalam-English

This dataset is obtained from 2 shared tasks: Offensive Language Identification in Dravidian Languages-EACL 2021 [27], and HASOC-DravidianCodeMix [31]. These tasks aim to identify offensive language from a code-mixed dataset of comments/posts on social media. The dataset contains all types of codemixing, such as inter-and intra-sentential. The mixing even happens at the script level, where some comments are written in Malayalam script, and others are in roman script. The details of the dataset are given in table 5.1.

Tamil-English

This dataset is obtained from the shared task: Offensive Language Identification in Dravidian Languages-EACL 2021 [27], and HASOC-DravidianCodeMix [31]. The Tamil-English code-mixed dataset was collected from tweets and comments. This dataset has comments/posts in roman characters. The statistics of the dataset are given in table 5.1.

Kannada-English

The dataset is obtained from the paper "KanCMD: Kannada CodeMixed Dataset for Sentiment Analysis and Offensive Language Detection". [71]. The dataset is collected from 18 youtube videos on different topics ranging from movie trailers, current trends about the ban on mobile apps in India, the India-China border issue, Mahabharata, and Transgenders. The data was collected between February 2020 and August 2020. The statistics of the dataset are given in table 5.1.

Bohra Hindi-English

The paper [16] created a corpus on hate-speech Hindi-English Code-mixed data. For each word, a language identification tag was assigned. The tags were 'eng', 'hin', and 'other'. 'eng', and 'hin' tag was assigned to words that are present in the English language and the Hindi languages, respectively. The 'other' is assigned to symbols, emoticons, hashtags, punctuations, named entities, and URLs. The data statistics are given in table 5.1.

HEOT Hindi-English

The paper [108] introduced a novel tweet dataset, Hindi-English Offensive Tweet (HEOT) dataset. It consists of tweets in Hindi-English code switched language split into three classes. The data statistics are given in table 5.1.

Telugu-English

The paper [137] created code-mixed Telugu-English corpus consists of 3361 Tweets. Each one of the tweets is manually annotated with one of the following tags: Overtly Aggressive (OAG), Covertly Aggressive (CAG), and Not Aggressive (NAG). Each token is annotated with a language tag. One of the following tags is assigned for language: eng, tel, and other, where eng stands for English, tel for Telugu, and 'other' for punctuation marks, emoticons, named entities, URLs, hashtags, etc. 'eng' is assigned to English words such as cold, good, etc., and 'tel' is assigned to Telugu words transliterated in English such as nenu, enduku', etc.

5.3 Experimental settings

In this section, we outline the experimental settings used in each of the models in the experiment.

We evaluated our model on a held-out test dataset for all experiments. We evaluate the baseline models on the following metrics: macro-F1, weighted-F1, and accuracy. In comparison, we evaluated the shared task models on the weighted-F1⁹. We chose macro-F1 because they take the imbalanced data into consideration and give an idea of whether the model is fine-tuned toward the labels with more data points. The macro-F1 method can be used to know how the system performs overall across the sets of tags in the dataset. In weighted F1, the F1 Scores are calculated for each label, and then their average is weighted by the number of true instances for each label. It is used to emphasize the importance of some samples. Accuracy-score helps to determine how accurate the model is while predicting the results.

During pre-processing, we have used a deep transliteration tool known as ai4bharat-transliteration¹⁰ library. The multilingual pre-trained BERT is obtained from the transformers library [169] and is fine-tuned for the hate speech and offensive language detection task. The optimizer used is weighted Adam with the learning rate of 2e - 5 and epsilon value equal to 1e - 8. The loss function used is a cross-entropy loss. The number of epochs used for training the model is 30 with batch size set to 128.

For training the CM RoBERTa model, The maximum truncation length is kept as 512. The min_frequency of words for the tokenizer is 1. The learning_rate is 1e - 5, the number of train epochs is 5, and the GPU training batch size is 8.

5.4 Baselines

In this section, we describe the baselines that we explore in the context of hate speech detection code-mixed text.

⁹As all the models from the shared task are evaluated using the weighted-F1 score

¹⁰https://pypi.org/project/ai4bharat-transliteration/

5.4.1 Support Vector Machines

We chose Support Vector Machines (SVM) [41] for hate speech and offensive content detection. The SVM is a max-margin classifier that segregates two classes with a hyperplane.

5.4.2 FastText

The code-mixed text contains a primary language called the matrix language, as well as a secondary language, i.e., the embedded language. In order to represent code-switching text, the straightforward idea is to train the model with word embeddings. We used the FastText model [64] to train the word embeddings. FastText has a subword-level word representation feature. Therefore, we utilized subword-level embeddings from FastText to solve variational representations of spellings and translations. Finally, we used the fastText classifier to detect hate speech and offensive language in the given code-mixed data.

5.4.3 CNN

[90] This convolutional neural network-based text classifier is trained by considering pre-trained word vectors.

5.4.4 Bi-LSTM

[76] A two-layer, bi-directional LSTM text classifier with pre-trained word embeddings [64] as input was considered for the task of text classification.

5.4.5 Multilingual BERT (mBERT)

Multilingual BERT [46], is pre-trained on Wikipedia text across 104 languages. The pre-trained model uses the Transformer architecture [157]. It has shown a surprising multilingual ability to outperform existing strong models, either multilingual or code-mixed/code-switched on multiple zero-shot cross-lingual tasks [121, 173].

5.4.6 XLM-RoBERTa (XLMR)

[38] is a multilingual language model. It is pre-trained in 100 languages. It obtained a state-ofthe-art performance on cross-lingual classification, sequence labeling, and question answering. XLM-ROBERTa improves upon BERT, such as training on a larger dataset, dynamically masking out tokens compared to the original static masking, and using a Byte-Pair-Encoding as the pre-processing technique and a dual-language training mechanism with BERT in order to learn better relations between words in different languages.

5.4.7 Genetic Algorithm

[136] presented a genetic algorithm technique by ensembling CNN, XLMR-Base, and a custom-pretrained XLM-Roberta-Base Classifier. The genetic algorithm performed the best on the given Dravidian code-mixed Tamil and Malayalam datasets.

5.4.8 AWD-LSTM with mBERT + XLMR

[87] presented an ensemble of an AWD-LSTM-based model along with multilingual BERT and XLM-RoBERTa. Their work also incorporates additional training data obtained from the monolingual English Offensive Language Identification Dataset (OLID) [180] used in the SemEval-2019 Task 6 (OffensEval) [181].

5.5 Performance

Table 5.2 shows the experimental results in terms of weighted f1 obtained on the code-mixed hate speech datasets. We observe that the proposed approach of converting code-switched data into respective high-resource languages outperform the baselines significantly. A hate speech dataset with more training instances, i.e., the Tamil-English and Telugu-English dataset, performed better on the proposed approach than the other datasets.

On the original dataset, the models SVM, Bi-LSTM, FastText, and CNN didn't show much improvement, whereas the transformer models have vastly improved because of their pre-trained embeddings. The SVM didn't perform well on the dataset because it considers only the syntactic information to extract features from the words. The naive neural network models such as Bi-LSTM, FastText, and CNN didn't perform better. A reason might be the model's ineffectiveness in capturing the semantics of the words in their embeddings. On the other hand, the multilingual BERT model performed better than the baselines due to its state-of-the-art transformer architecture, its effectiveness in capturing better pretrained embeddings, and its state-of-the-art sub-word tokenization technique. The XLMR model gave a competitive performance to the mBERT model on all the datasets.

As shown in table 5.2, our model performed better when compared to the baselines because of its fusion learning of code-mixed and mBERT models. In this fusion learning approach, if a data point's tag depends on the semantic representation of the pre-trained embeddings, then the feed-forward network selects the mBERT model. On the other hand, when the pre-trained embeddings didn't satisfy the given data point, the feed-forward network was inclined towards the custom code-mixed model.

We have seen that including the profanity vector improved the performance of the HEOT Hi-En[108] dataset. Because in the HEOT Hi-En dataset, the hate speech words are manually listed. Whereas for other datasets, we have seen a slight decrease or no-change scenarios in classification metrics. The reasons are as follows.

Data/Model (Weighted-	Bohra Hi-En	HEOT Hi-En	Ml-En	Ta-En	Kn-En	Te-En
F1)						
SVM	62.8	84.5	88.3	72.7	48.4	69.1
SVM + PV	62.7	85.2	87.8	71.9	47.8	69.7
FastText	63.1	84.6	89.7	73.3	49.5	86.5
FastText + PV	63.1	84.8	89.5	73.1	49.5	86.3
CNN	63.2	83.1	89.9	73.5	70.1	91.9
CNN + PV	63.1	83.5	89.4	72.5	69.7	91.5
Bi-LSTM	64.5	84.7	88.6	73.1	49.4	92.4
Bi-LSTM + PV	63.8	85.1	88.3	72.9	49.1	92.1
mBERT	67.1	86.2	97.2	74.5	73.1	95.5
mBERT + PV	66.2	86.2	96.4	74.0	72.5	95.1
XLMR	66.8	85.3	95.3	74.1	71.8	94.9
XLMR + PV	65.9	85.5	94.8	73.1	71.5	93.6
Genetic Algorithm [136]	68.9	86.1	97.3	78.3	74.1	96.1
AWD-LSTM with	68.5	86.2	97.0	77.4	72.3	95.6
mBERT + XLMR [87]						
Our Approach -	67.5	85.4	95.0	74.7	71.2	95.5
mBERT with data						
augmentation						
Our Approach -	67.3	86.1	96.3	74.9	64.8	95.5
mBERT with class						
balanced loss [53]						
Our Approach - Fusion	68.7	86.2	97.2	77.4	74.1	96.1
Model [56]						

Table 5.2 Weighted-F1 score on the baselines without and with profanity vector (PV) when compared to our approaches on the hate speech datasets

1. The change of spelling and script in the code-mixed datasets and the presence of homo-graphic words did attribute to incorrect translation. Such words were either unidentified or misclassified as hate words during profanity word identification.

- 2. The translators are not perfect. The imperfections in translations resulted in unidentified/misclassified hate words.
- 3. The profanity word identification works at the word level. But there might be phrasal-level hate speech words that were not identified by our method.
- 4. The negative sarcasm words are not identified by our method.
- 5. The complexity of the models and the attention mechanisms used in the models already helped the supervised models to capture the related hate words/phrases, which helped the model for better hate speech detection even without profanity vector identification.

Given the code-mixed dataset, Profanity word identification is a difficult task due to script changes and translation issues, some misclassified and unidentified hate words might decrease the model's performance. However, correctly identifying all hate speech words is beneficial to the model, as seen in the HEOT Hi-En dataset.

We know that emoticons have implicit meaning in hate speech detection. We have used the python emoji¹¹ library to identify the emoticons and convert them to text. But this conversion did not improve the results on these datasets. Because of the following reasons.

- 1. We have observed that the text already has hate-inducing words followed by emoticons.
- 2. There was no substantial text that was hate-inducing because of emoticons.

The conversion of emoticons might help in other datasets with a substantial amount of hate-induced data due to emoticons.

5.5.1 Ablation study

We have conducted an ablation study of our approach on the given datasets. In the ablation study, we have removed individual components from the model to determine our model's dominant component and verify whether our model supports graceful degradation. First, we have removed the profanity vector augmentation from the proposed method. Second, we have removed pre-processing step to determine the effect of transliteration and correction of variations in spellings on the model. Third, We have removed the mBERT model to assess the impact of pre-trained embeddings on our model. Finally, we have removed the CM model to determine the effectiveness of including the CM language model in our method. The details of the Ablation study are given in Table 5.3.

We can see that the mBERT model was the more dominant model in our method, and removing the mBERT model has shown a great decrease in accuracy. Further degradation in the accuracy is shown by removing both the pre-processing step and the mBERT model. It is because the transliteration helped obtain better embeddings for the mBERT model, which boosted the model's accuracy.

¹¹https://pypi.org/project/emoji/

Ablation Study	Bohra Hi-En	HEOT Hi-En	Ml-En	Ta-En	Kn-En	Te-En
Our Approach	69.1	86.1	96.8	78.9	75.6	96.1
Our Approach w/o pro-	69.1	86.1	96.8	78.9	75.6	96.1
fanity vector						
Our Approach w/o pre-	67.9	85.7	96.5	77.9	73.2	95.9
processing						
Our Approach w/o	66.4	84.8	96.4	78.0	71.9	91.6
mBERT						
Our Approach w/o CM	68.5	85.2	96.5	78.5	74.6	93.7
model						
Our Approach w/o	64.5*	84.6*	96.2*	77.6*	70.5*	89.5*
pre-processing and w/o						
mBERT						
Model w/o pre-	68.0	84.9	96.5	78.3	73.8	93.7
processing and w/o						
CM model						

 Table 5.3 The ablation study of our model by using "accuracy score" as the investigation metric on the code-mixed hate speech datasets.

The removal of the profanity vector didn't disturb the model's accuracy because of the state-of-theart learning mechanism of the vanilla transformer model that can learn the semantics of the words while fine-tuning the dataset. However, the profanity vector gives the benefit of early loss minimization, which will be described in detail in section 5.5.3.

We know that code-mixed languages are of low resource. Low-resource languages often suffer from a lack of labeled data for training hate speech detection models. Building a comprehensive dataset for such languages can be challenging. However, profane word lists can provide a starting point for identifying offensive language in the absence of extensive training data. By using a profane word list as a baseline, you can establish a basic framework for detecting explicit profanity, even with limited labeled examples. This work presents the first approach for automated profane word list identification. This word list contains false positive profane words. The use of advanced algorithms for identifying profane words will benefit the performance of model and is left for future scope.



Figure 5.4 Comparison of mBERT baseline and Our Approach on Hindi-English datasets

5.5.2 Confusion matrix

The confusion matrix is a tabulated layout that allows better visualization of the performance of a supervised model to get a better idea of the model performance. We can see whether the system is



Offensive Targeted Insult I eted Insult Grou Offensive_Targeted_Insult_Other

Fee

Elle



(e) mBERT on Malayalam-English

(f) Our approach on Malayalam-English



mislabeling the classes in the given classification dataset through the confusion matrix. In our matrix, each row represents the instances in an actual class, while each column represents the instances in a predicted class.

From the table 5.2, we can see that the baseline model "mBERT" gave a competitive performance to the proposed model. So we deeply investigated the two models in terms of their prediction based on the confusion matrices. The matrices are given in the figures 5.4 and 5.5.

From the figures, our model shows an improvement in the hate speech labels compared to the mBERT model. Our approach didn't overfit itself to the highly labeled datasets. Our model made a balanced decision to get the correct labels across most of the classification label instances. This scenario can be seen in the Dravidian datasets (Tamil-English, Kannada-English, and Malayalam-English). The model also tried to predict the scarcely tagged labels (Offensive_Targeted_Insult_Individual, Not-{Tamil/Kannada/malayalam}, Offensive_Targeted_Insult_Group, Offensive_ Untargetede) in the Dravidian datasets and achieved a better accuracy when compared to the mBERT model. This shows the efficiency of our approach when compared to the mBERT model. It proves that our model helps better detect hate speech when compared to the current state-of-the-art baselines.

5.5.3 Training Loss

Loss is the penalty for a bad prediction. Training loss is the error in the training set of data. The training loss helps us determine the model's tuning toward the datasets. If the fine-tuning is good, then the model is said to perform better on the given test set too. The training loss on the proposed model (which includes profanity vector), our model without profanity vector) and its closest competitor, the mBERT model, is given in Figure 5.6. We can see the proposed approach has shown a better decrease in loss when compared to the mBERT model. It is due to using a custom code-mixed model in combination with the mBERT model. The profanity vector has also added additional linguistic modeling of data helping the model to perform better on the given training dataset. Using the profanity vector, the proposed model obtained a minimized loss during the initial epochs compared to the mBERT model. However, all the approaches reached a similar optimum due to the dominance of the mBERT model. After reaching local optima, we observed a minute decrease in training loss for the given datasets. Out of all the datasets, our model has shown a better fine-tuning capability on the Tamil-English dataset. The number of instances in the Tamil-English dataset helped the code-mixed language model to fine-tune the data better. On the other hand, the Malayalam-English dataset has shown us a low deviation in training loss between the mBERT and the proposed approach. It is because the dataset is highly inclined towards Not_offensive tags. This imbalance credited the model as even the baselines performed better on that dataset.



(a) Loss on Bohra Hi-En dataset





(b) Loss on HEOT Hi-En dataset



(d) Loss on Malayalam-English Dataset

(c) Loss on Kannada-English dataset



(e) Loss on Tamil-English dataset

Figure 5.6 Comparison of Training loss of our approach (with or without profanity vector) and the mBERT model on the hate speech datasets

5.6 HSD in Dialog System

We used the proposed HSD model trained on the Telugu-English code-mixed hate speech dataset for automated hate speech detection in our dialog system.

During the interaction, our task-oriented dialog system continuously monitors each utterance that the user has entered to identify/detect hate speech and abusive content. If such hateful comments are identified, the session is immediately terminated.

By incorporating this hate speech detection module, we aim to create a safe and inclusive environment where users can engage in constructive and respectful conversations. This proactive approach helps prevent the dialog system from processing and responding to hate speech, thus ensuring that offensive and harmful content is not perpetuated. By actively detecting and addressing hate speech, we strive to maintain a responsible and ethical dialog system that promotes positive interactions and fosters a respectful atmosphere for all users.

5.7 Summary

In this chapter, we presented our work on code-mixed hate speech detection. In our work, we developed deep learning models using BERT encoders for classifying code-mixed hate speech in Dravidian languages. We also addressed the imbalance in the data using class-balanced loss, and we proposed a better code-mixed model using a fusion of two models with profanity vector augmentation.

Chapter 6

Dialog Manager and Suggestion System

In this chapter, we present our work on Dialog manager and Suggestion system¹. These 2 modules play a significant role in the overall dialog flow between the patient and the system.

6.1 Dialog Manager

Dialog management is the process of tracking and controlling the conversion to direct the user toward a specific task. It is accomplished by the dialog manager. The dialogue manager is the major component of a spoken dialogue system. It accepts the interpreted input from the NLU. It interacts with external knowledge sources. It helps produce the necessary output for the user and generally controls the dialogue flow.

The dialog manager can be viewed as having two main tasks:

- 1. Dialog state, it tracks the state of the dialog
- 2. Dialog control makes decisions about the next system action.

The dialog state stores a record of what has been said so far in the dialog. It provides the information used for dialog control. Dialog control describes the information to be gathered in the dialog. It holds the information about one or more slots to be filled with values provided in the course of the dialogue.

The Dialog Manager module can be implemented in several ways using

- 1. Dialogue scripts
- 2. Frames and slot filling
- 3. Statistical dialog manager

¹Suggestion system was published in the WiML 2022 workshop NeurIPS and is supported by "Touchstone" KCIS Enabler Grant Project, IIIT Hyderabad

Dialogue scripts define the actions that can be taken at each point (or state) of the dialogue. This method is also known as finite-state dialogue control. The predicted dialog states and the transitions between them can be depicted in a graph, which can be represented diagrammatically. This approach is used for system-directed dialogues.

Frame-based dialogue Manager is more flexible. Frames consist of slots that are filled with values elicited from the user. The aim is to fill these slots by gathering user values and retrieving records from a database that satisfies the user's constraints. This type of dialogue has also been referred to as form-filling. Even though it looks similar to dialog script, the added value of frame-based control is that it supports more flexible dialogues by allowing the user to fill in the slots in different orders and combinations. For example, if the user responds to the system's initial prompt with more than one value, all the respective slots obtained from the user's response are filled, and the system can then ask for values for any slots that remain open. Thus the flow of the dialogue is not predetermined but depends on the information the user provides in response to the system's prompts.

Statistical dialog Manager: This approach uses statistical models, such as machine learning algorithms, to learn patterns in the user's input and generate appropriate responses. This approach requires a large amount of training data and can be more flexible than rule-based systems.



Figure 6.1 Patient's frame to be tracked and updated by the dialog system via user interaction.

A Frame-Based Dialog Manager for Task-Oriented Dialog

Generally, the choice of dialogue management strategy is determined by the application. Task-based dialogue systems are usually implemented using a form-filling approach. A dialog model is a form with slots to be filled, and the dialogue control is mostly system-directed.

The dialog scripts require careful design of rules, strategies, and dialogue models. However, the major problem is that it is difficult to design all the necessary rules to cover all aspects of dialogue management, particularly when considering the uncertainties that pervade at every level of dialogue. An alternative is to use a data-driven method involving statistical machine learning, such as supervised learning.

Supervised learning can be used when there is a vast corpus of examples of the optimal decisions to make in dialogue. From such a labeled corpus, a stochastic/probabilistic dialogue model can be learned (or estimated). Then, it can be applied to new dialogues similar to those recorded in the corpus.

But given our limited task-oriented corpus and its complexity in doctor-patient interactions, we chose to use a frame-based dialog manager by writing handcrafted rules to track and update the frame. Our frame looks similar to a patient's profile given in figure 6.1. The dialog manager will ask a series of questions until the given frame is completely full.

The frame-based approach has several advantages over other dialog management techniques. First, it allows for a more flexible and adaptable system, as the frame can be updated dynamically based on the user's input. Second, it allows for a more natural and intuitive conversation, as the system can take into account the user's goals and preferences when generating responses.

6.2 Suggestion System

When the user enters the patient's information, it might be possible that the user fails to mention a few symptoms. So there must be a suggestion system to remind the user if they have any relevant symptoms. An example of the doctor suggesting symptoms to a patient during an interaction is given in Figure 6.2.

The input data available to the suggestion system are the user's symptoms and clinical data. So the suggestion system must use clinical data and the user's symptoms to suggest the most relevant symptoms. Figure 6.3 presents an abstract representation of the suggestion system module.

The underlying assumption is that the functionality of the human body is similar in most people. Therefore, the symptoms exhibited for a medical condition will also be similar. By obtaining similarities between the symptoms, we can suggest the relevant symptom(s) to the user.

This phenomenon of suggesting the most likely symptoms to the user by collaborating the symptoms from similar-patients² in the clinical data is called collaborative filtering (CF). To be specific, it comes under content-based collaborative filtering.

²The patients who have exhibited similar symptoms entered by the user.

Speaker	Utterance			
d	వయస్సు? (age?)			
р	25		Ļ	Patient presenting their
	sir ముక్కు అంతా మొత్తం మూత పడ్డట్లు అయింది. తుమ్ములు, చీమిడి			symptoms.
р	(Sir nose is completely blocked, Sneezing and mucus.)	_	J	
р	శ్వాస ఆడుతలేడు. (I am unable to breath)			
d	ఎన్ని రోజుల నుంచి? (Since, How many days?)			
р	15, 20 రోజులు అయితాంది (Since, 15, 20 days.)			
d	ఆహా (ok)			Doctor Suggesting
d	దగ్గు వస్తుందా? (Do you have cough?)		\geq	patient might
р	దగ్గు లేదు. (No I don't have cough.)			have
d	తలనొప్పి? (Head ache?)			
	తలనొప్పి, నర నరం గుంజుకొస్తాంది. (I have head ache, every nerve			
р	is paining)		J	

Figure 6.2 Doctor suggesting symptoms to a patient during an interaction.



Figure 6.3 Suggestion System Model.

Content-based user collaborative filtering is a recommendation system approach that combines the ideas of content-based filtering and collaborative filtering. In this approach, the system uses the user's past symptoms to generate a user profile and uses this profile to recommend symptoms by collaborating the symptoms from similar patients.

User	symptoms list
U_1 :	[fever, cold]
U_2 :	[fever, cold, cough]
U_3 :	[blister, skin]
U_4 :	[rash, skin]

Table 6.1 Symptoms selected from four interactions in the clinical data

For content-based collaborative filtering, we need efficient models to extract features for each conversation from the raw conversational data. So we relied on various feature extraction models to get features. The SOTA models considered for feature extraction are TF-IDF, SVD, RNN, CNN, mBERT. Later on, we used user-based and item-based collaborative filtering to suggest relevant symptoms to the current user.

User-based collaborative filtering uses the behavior and preferences of similar users to generate recommendations for the current user. To implement user-based collaborative filtering, the system typically creates a user-symptom matrix, where the users are the rows of the matrix and symptoms are the features/columns of the matrix. It then calculates the cosine similarity between the target user and other users w.r.t their symptoms. Once the system has identified the most similar users to the target user, it uses the information from similar users to generate suggestions.

Item-based collaborative filtering uses the similarity between items to generate recommendations for a given user. To implement item-based collaborative filtering, the system typically creates a user-symptom matrix, where the users are the rows of the matrix and symptoms are the features/columns of the matrix. Later on, the system typically creates an item-item matrix that contains the similarity between all pairs of symptoms. It then uses the symptoms that the user already has and finds the most similar symptoms based on the item-item matrix. Finally, It recommends similar symptoms to the user.

An example that puts forth the behavior of suggestion is provided below.

Table 6.1 shows symptoms selected from 4 conversations in clinical data (where U_n refers to a user conversation).

The one-hot vector space model of the example data provided in table 6.1 is represented in the useritem matrix 6.1.

(symptom/U)	blister	cold	cough	fever	rash	skin
U_1	0	1	0	1	0	0
U_2	0	1	1	1	0	0
U_3	1	0	0	0	0	1
U_4	0	0	0	0	1	1 ,

The TF-IDF vector will be of the form as represented here in the user-item matrix 6.2.

symptom/U	blister	cold	cough	fever	rash	skin
U_1	0	0.7	0	0.7	0	0
U_2	0	0.5	0.6	0.5	0	0
U_3	0.7	0	0	0	0	0.6
U_4	0	0	0	0	0.7	0.6

Item-based similarity finds the similarity between the items, is represented in the matrix 6.3.

(item/item	blister	cold	cough	fever	rash	skin
blister	1	0	0	0	0	0.7
cold	0	1	0.6	1	0	0
cough	0	0.6	1	0.6	0	0
fever	0	1	0.6	1	0	0
rash	0	0	0	0	1	0.7
$\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ $	0.7	0	0	0	0.7	1 /

As we don't want to recommend the same symptoms, we subtract the Identity matrix from item-item matrics.

When the current user (U_c) in an interaction enters "I have a fever and cold", the suggestion system considers both the symptom vectors of "fever", "cold" and aggregates them to suggest the relevant symptoms.

$$\begin{pmatrix} item/item & blister & cold & cough & fever & rash & skin \\ cold + fever & 0 & 1 & 1.19 & 1 & 0 & 0 \end{pmatrix}$$
(6.4)

The suggestion system takes the topmost symptom. In our case, the suggestion system suggests "cough".

If the user (U_2) enters "I have skin problems", the suggestion system suggests "blister", "rash". For user-based similarity, the U_c constructs the user_item matrix of the form,

$$\begin{pmatrix} item/item & blister & cold & cough & fever & rash & skin \\ U_c & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$
(6.5)

Then, we calculate the cosine similarity between the current users and all the other users in the dataset. The obtained cosine similarity between the users is

$$\begin{pmatrix} U/U & U_1 & U_2 & U_3 & U_4 \\ U_c & 1 & 0.8 & 0 & 0 \end{pmatrix}$$
(6.6)

We can see that U_1 and U_2 have a high similarity with the current user U_c . Thus, we extract all the symptoms from the users U_1 and U_2 such as *fever*, *cough*, *and cold*. As we need to suggest new symptoms, we ask if the user has "cough".

6.3 Content Based Collaborative Filtering Models

6.3.1 TF-IDF based collaborative Filtering

TF-IDF based collaborative filtering [117] is a recommendation system approach that combines the concepts of TF-IDF (Term Frequency-Inverse Document Frequency) and collaborative filtering. In this approach, the system uses the TF-IDF algorithm to construct a user-item matrix where the rows are each user's conversations and items are symptoms/medical conditions. Then, it uses user-based and item-based collaborative filtering to recommend the most similar symptoms to the user.

6.3.2 SVD-based Collaborative Filtering

SVD has been studied a lot in recommendation systems. [138] used latent semantic analysis based SVD approach to extract similarity between two or more symptoms. It uses a matrix structure where each row represents a user, and each column represents an item. This matrix's elements are the ratings given to items by users. $A = USV^T$. Where A is a utility matrix, U is an orthogonal left singular matrix, which represents the relationship between users and latent factors, S is a diagonal matrix, which describes the strength of each latent factor, and V is a diagonal right singular matrix, which indicates the similarity between items and latent factors. The latent factors here are the characteristics of the items. The SVD decreases the dimension of the utility matrix A by extracting its latent factors. It maps each user and each item into an r-dimensional latent space. This mapping facilitates a clear representation of relationships between users and items.

But the problem with the above approach is that they capture only linear representations. They won't work if the data has a complex or deep relationship between two symptoms. So we move on to deep learning models to extract the relationship between the symptoms.

6.3.3 Collaborative filtering using Deep Learning Methods

We have used the Feed Forward Networks, RNNs, CNN's, and transformer Models to extract a more fine-grained relationship between two symptoms. The masked language model is portrayed as a recommendation system. Here, we have used the masked language modeling technique and trained the NN model language model to predict the masked symptom. By fine-tuning the model on the task of masked language modeling, the model will learn to predict the correct recommendation based on the given input.

6.4 Evaluation of Suggestion system

Table 6.2 shows the performance of the five different models (TF-IDF, SVD, RNN, CNN, and mBERT) on a content-based recommendation system with collaborative filtering on a dataset using

Models	User Based			Item Based		
	MAR@10	MAP@10	F1-score	MAR@10	MAR@10	F1-score
TF_IDF	0.17	0.01	0.03	0.40	0.18	0.24
SVD	0.19	0.02	0.04	0.35	0.17	0.22
RNN	0.12	0.01	0.02	0.30	0.16	0.20
CNN	0.17	0.02	0.03	0.42	0.19	0.25
mBERT	0.16	0.03	0.05	0.65	0.23	0.29

 Table 6.2 Performance of the content-based recommendation systems with collaborative filtering on our dataset.

user-based and item-based approaches. The evaluation is based on two metrics: mean average recall (MAR) and mean average precision (MAP) at 10 recommendations.

For the user-based collaborative filtering approach, we can see that SVD has the highest MAR@10 score of 0.19, indicating that on average, 19% of the relevant recommendations are included in the top 10 recommendations. However, for MAP@10, mBERT has the highest score of 0.03, indicating that, on average, the relevant recommendations are ranked higher in the top 10 recommendations.

For the item-based collaborative filtering approach, the table shows that mBERT outperforms the other models in both MAR@10 and MAP@10, with scores of 0.65 and 0.23, respectively. This indicates that mBERT is able to recommend a higher percentage of relevant items in the top 10 recommendations and also rank them higher on average compared to the other models.

We can say that the deep learning-based suggestion systems gave better performance when compared to the other methods. Deep neural networks are capable of modeling the non-linearity in data with nonlinear activations. This property makes it possible to capture the complex and intricate user item or item-item interaction patterns when compared to linear models such as matrix factorization, cosine similarity, and kNN models.

Even though we performed an elaborate study on content-based recommendation systems using various SOTA models and collaborative filtering approaches, we found that the results were still low on various models. Because a single symptom is expressed differently, i.e., using various suffixes and prefixes in the Indian language.

Compound agglutinative Indian languages present unique challenges in computational processing due to their complex morphology and syntax. Telugu, in particular, has a rich inventory of inflectional and derivational suffixes that can be added to words to convey a wide range of meanings. This leads to an explosion in the number of possible word forms, making the computational processing of Telugu more challenging than other languages with simpler morphological systems. This can be explained using a simple example 6.3.

Symptom	Different variations of same symptom							
	muMxara hair aMwA loss ayipoyiMxi. (hair loss on the front)							
	hair fall							
	hair loss							
	hair problem uMxi (hair problem is present)							
	hair lightgA rAlipowA uMxi.(light hair fall)							
	hair damage <i>uMxi</i> lightgA (light hair damage)							
	unnaxi UdipowuMxi (what is present, is going away)							
	hair damage uMxi. (hair damage is present)							
	hair fall <i>ayipowuMxi</i> (hair fall)							
	koVMcaM hair fall (slight hair fall)							
hair fall	hair loss <i>avuwuMxi</i> (hair fall)							
	hair fall <i>avuwuMxi</i> (hair fall)							
	winnAgA rAlipoVwuMxi (slightly falling)							
	rAlipowuMxi. (falling)							
	80 nuMci 100 veVMtrukalu UdipovadaM (80 to 100 count hair fall)							
	<i>eVkkuva</i> hair fall (high hair fall)							
	hair fall <i>uMxi</i> (hair fall is present)							
	hair fall, juttu rAlipoVvadaM anexi waggadaM lexu (hair fall is not decreasing)							
	juttu powaxi (hair fall)							
	hair fall, hair <i>rAlipoVwuMxi</i> . (hair fall)							
	hair loss, hair fall							

 Table 6.3 Variations of single symptom

The above example in Table 6.3 shows that a single symptom is expressed in different formats. And we have only shown a single example of the above symptom. But there are thousands of such symptoms, having implicit meanings in the dataset. Writing rules and using machine translation won't work in such scenarios. This explosion of variations makes the computational processing of the Telugu language a very difficult task. Thus it is difficult to normalize such symptoms to one. Such scenarios resulted in reduced performance of the recommendation system.

6.5 Summary

This chapter elaborates on dialog management and suggestion system. The dialog manager is considered the heart of the dialog system, which directs the user toward a specific task. We used a frame-based dialog manager for our task-oriented dialog system. We then present our work about the suggestion system, which reminds the user if they have any relevant symptoms. We used collaborative filtering methods for the suggestion system.

Chapter 7

Medical Specialist Classification

In this chapter, we present our work on medical specialist classification, i.e., to predict the medical specialist based on the data entered by the user. ¹.

Medical diagnosis involves a complex process of analyzing patient symptoms and medical history to arrive at a correct diagnosis. This task is usually performed by a medical specialist who has a deep understanding of the specific medical condition. In recent years, there has been a growing interest in using machine learning techniques to aid medical diagnosis and prediction tasks. The task-oriented dialog systems that can classify the appropriate medical specialist based on a patient's symptoms and medical history are promising applications of such techniques [107, 101].

Several studies have been conducted on task-oriented dialog systems in healthcare. In [88], a conversational system was developed to assist seniors with their medication. In [105], a conversational system was developed to assist in the early detection of Parkinson's disease. In [112], a conversational system was developed to assist in the management of diabetes. However, to the best of our knowledge, no studies use code-mixed datasets for medical specialist classification.

7.1 Classification Models

In this chapter, we use a deep learning classifier for predicting the right specialist based on the obtained medical information from the user and our code-mixed TOD dataset. As discussed in the earlier chapters, we have already gone through the methods that were used to interact with the user to obtain medical information such as symptoms, medical history, and other relevant information. The dialog system finally uses this information to predict the appropriate medical specialist to consult.

Therefore, in order to predict the correct medical specialist, a trained classifier is required. Also, We know that each interaction in our dataset is labeled with the appropriate medical specialist. We utilize the labeled conversations in the dataset to train a classification model. Also, we know that all the conversations are pretty long and contain unnecessary information. However, as mentioned in our slotfilling model, our slot-tagging model extracts the necessary medical information required for specialist

¹This work is supported by "Touchstone" KCIS Enabler Grant Project, IIIT Hyderabad

classification. Thus we used the slot tagging model to extract the necessary medical information and used it to classify the specialist.

We explored several state-of-the-art text classification models such as Extra Trees Classifier (ETC), CNN, Bi-LSTM, fastText, and mBERT and its variants for medical specialist classification. The goal was to identify the best-performing model for predicting the appropriate medical specialist. We selected the ETC machine learning model as it recorded the best performance compared to all the machine learning classifiers for the given dataset.

The state-of-the-art classification models considered are,

7.1.1 Extra Trees Classifier(ETC)

ETC is a machine learning algorithm that belongs to the family of decision tree-based ensemble methods. ETC builds an ensemble of decision trees using a random subset of features and training samples at each node, resulting in a forest of decision trees that are independently grown and combined to make the final classification decision. Unlike a Random Forest Classifier, which selects the best split among a set of randomly selected features, ETC selects the split point randomly, which can lead to faster training times and better generalization performance. Another advantage of ETC is that it can handle imbalanced datasets well. It can give more weight to minority classes in the training process, leading to better classification results for rare classes. ETC is also computationally efficient and can handle large datasets with high-dimensional feature spaces.

7.1.2 CNN

[91] In this paper, the author introduced a model called the "Convolutional Neural Network for Sentence Classification" (CNN-S). The model takes a sentence as input and uses a one-dimensional convolutional layer with multiple filter sizes (3,4,5) and max-pooling to capture the local features of the sentence. The output of the convolutional layer is then fed to a fully connected layer, followed by a softmax activation function for text classification.

7.1.3 **Bi-LSTM**

[79] In this paper, the authors proposed a new model for sentence classification that uses a BiLSTM to capture the contextual information of a sentence. The model takes a sentence as input and feeds it through a BiLSTM layer that processes the sentence in both forward and backward directions. The output of the BiLSTM layer is then fed to a fully connected layer followed by a softmax activation function for text classification.

7.1.4 fastText

[82] This is a simple, efficient, and fastest text classification method. It treats the average of word/ngrams embeddings as document embeddings, then feeds document embeddings into a linear classifier. The paper presents several modifications to the traditional bag-of-words model, including the use of subword information, character-level features, and the hashing trick, which were proven very beneficial for text classification on utf-8 encoded languages.

7.1.5 Text GCN

[178] Builds a heterogeneous word document graph for a whole corpus and turns document classification into a node classification problem. The approach involves representing a sentence as a graph, where each word is represented as a node, and the relationships between words are represented as edges. The GCNs are then used to perform convolution operations on the graph, allowing the model to capture the dependencies and interactions between words in the sentence. It uses GCN [92] to learn the word and document embeddings. One of the key contributions of this paper was the introduction of the use of GCNs for text classification, which allowed the model to capture the dependencies and interactions between words in a sentence and improve its performance on text classification tasks.

7.1.6 mBERT

The BERT model [46] The mBERT is based on transformer architecture [157] It uses a series of multiheaded attention and feedforward networks. It has been pre-trained on a large corpus in 104 languages, including low-resource languages. The model is based on the Transformer architecture and is trained using a masked language modeling objective, where a certain percentage of the words in the input text are randomly replaced with a special mask token, and the model is trained to predict the original words. We have also considered IndicBERT [85] and Clinical-BERT [6] model. With IndicBERT being trained on a large-scale corpus, covering 12 Indian languages and Clinical-BERT model is trained on clinical notes to obtain embeddings specific to medical data respectively.

7.1.7 XLM-R

It is a transformer-based language model and is pre-trained on large amounts of monolingual and parallel corpora in over 100 languages and can effectively transfer knowledge from high-resource languages to low-resource ones. The embeddings produced by XLM-R have been shown to be effective for a wide range of cross-lingual transfer learning tasks, such as cross-lingual text classification, cross-lingual named entity recognition, and cross-lingual question answering. One notable feature of XLM-R is its ability to handle code-switching, which is common in multilingual text. XLM-R can process text with mixed languages and provide effective embeddings for downstream tasks.

Model	Precision	Recall	F1-score	Accuracy
ETC	0.85	0.88	0.86	0.86
CNN (with FastText embed)	0.87	0.88	0.87	0.88
CNN (with BERT embed)	0.87	0.88	0.87	0.88
Bi-LSTM (with FastText embed)	0.89	0.90	0.89	0.89
Bi-LSTM (with BERT embed)	0.89	0.90	0.89	0.89
FastText	0.88	0.89	0.88	0.89
Text GCN	0.88	0.89	0.88	0.89
Clinical-BERT	0.89	0.90	0.89	0.89
indic-BERT	0.92	0.91	0.92	0.92
mBERT	0.93	0.92	0.92	0.93
XLM-R	0.92	0.92	0.92	0.92

Table 7.1 Performance of text classification models on the code-mixed medical dataset.



Figure 7.1 F1-score for each specialist label when mBERT classifier is used.

7.2 Experimental settings

We implemented neural network models using the Pytorch library. The huggingface-transformers library is used to obtain pre-trained BERT models. We used multilingual fastText embeddings [64] of dimensionality 300 for CNN, Bi-LSTM, fastText, and TextGCN models. We used default parameter settings as in their original papers for implementations. The train validation and test splits are achieved at


Figure 7.2 Confusion matrix for specialist classification task - mBERT

70%, 10%, and 20%. We evaluated the classifier using the weighted-F1, precision, recall, and accuracy classification metrics.

7.3 Performance

The performance of the baselines for the medical specialist classification on our dataset is given in Table 7.1. The models are evaluated based on precision, recall, F1-score, and accuracy metrics.

From the table 7.1, we can see that all the methods performed well for the classification task. This behavior is attributed to the NLU's slot-filling task. The NLU's slot-filling model performed well in extracting the slots related to the symptoms. This in-turn, benefited the classifier in getting essential information and thus associated with improved accuracy.

The best-performing model among all the models is mBERT, which achieved the highest precision, recall, F1-score, and accuracy metrics of 0.93, 0.92, 0.92, and 0.93, respectively. indic-BERT and XLMR also performed well and achieved high F1 scores of 0.92.

The results suggest that pre-trained language models such as mBERT, indic-BERT, and XLMR are effective in handling code-mixed medical text classification tasks. Bi-LSTM with BERT embeddings and Clinical-BERT also performed well but slightly lower than the best models. The results also suggest that using pre-trained embeddings such as FastText and BERT can improve the performance of CNN and Bi-LSTM models for text classification. Overall, the results demonstrate that deep learning models can effectively handle code-mixed text classification tasks, and pre-trained language models can significantly improve their performance.

Based on figure 7.1 The top performing specializations are Orthopaedist and Ophthalmologist, both with a weighted f1-score of 0.98. On the other hand, the lowest performing specialization is General Physician, with a weighted f1-score of 0.75. The mBERT model is evaluated using the confusion matrix, too, as shown in figure 7.2. From the confusion matrix, we can see that the model performed well on the dataset. The lowest performing specialization, "general physician", is often misclassified with other specializations. It is a typical phenomenon because some patients prefer consulting a physician before proceeding with other medical specialists.

7.4 Summary

This chapter presents our work on the medical specialist classification model. We trained various models on our code-mixed corpus and chose the best among them. As the conversations were pretty long, we used slot tagging to extract relevant information from the dataset. The extracted information is then used to train the classifier. We have seen that mBERT model gave better performance when compared to other models for this task.

Chapter 8

Natural Language Generation

In this chapter, we present our work on Natural Language Generation¹. The Natural Language Generation (NLG) module plays a crucial role in a typical task-oriented dialog system. It converts a system action into a final response in natural language. The generated response should adequately represent semantic dialog actions and it should be fluent in engaging the user's attention. It acts as the ultimate interface to interact with users and significantly impacts the user's experience.

8.1 Input to NLG

The input to the NLG module is the dialog state that contains the intent and the slot information given by the dialog manager module, through which a system response in natural language is generated. Specifically, dialog state is defined as intent I and slot-value pairs.

$$\mathcal{A} = [\underbrace{\mathbf{I}}_{\text{Intent}}, \underbrace{(s_1, v_1), \cdots, (s_P, v_P)}_{\text{Slot-value pairs}}]$$
(8.1)

Also, our analysis of clinical data revealed a style accommodation phenomenon at the language level, wherein the doctor would adapt their language to match that of the patient by switching between code-mixed, Telugu, or English. To accommodate such language preferences in our dialog system, we introduced a language identification token in the dialog state. The language identification token tracks whether the previous user response was in code-mixed, Telugu, or English. Below is the updated dialog state with the language identification token:

$$\mathcal{A} = \left[\underbrace{\mathbf{I}}_{\text{Intent}}, \underbrace{ID}_{\text{lang ID}} \underbrace{(s_1, v_1), \cdots, (s_P, v_P)}_{\text{Slot-value pairs}}\right]$$
(8.2)

NLG aims to translate the dialog state A into a natural language response $x = [x_1, ..., x_T]$, where T is the sequence length.

¹This work was published in WiML 2022 workshop NeurIPS and is supported by "Touchstone" KCIS Enabler Grant Project, IIIT Hyderabad

8.2 NLG Models

NLG is a vastly studied research area. The most common types of NLG models are, Template-Based NLG and Machine learning - Based NLG.

- 1. Template-Based NLG involves creating pre-defined templates that can be filled in with specific data to generate text.
- 2. Machine learning Based NLG (such as an encoder-decoder model) uses machine learning models to generate a textual response. The models are trained on large amounts of data and learn to generate text based on patterns in the data. Machine learning-based NLG is particularly effective for generating personalized and creative texts by learning from massive datasets.

8.2.1 Template-Based NLG

Template-based natural language generation (NLG) is a technique used to generate human-like text by filling in pre-defined templates with relevant information.

The template-extraction process consists of

- 1. Preprocessing the dataset to remove irrelevant information from the dataset
- 2. Identify the most common patterns and structures in the preprocessed dataset.
- Group similar patterns together and create templates based on them. Templates should include placeholders for variables that can be filled in with specific information based on the current dialogue context.
- 4. Validate the templates by manually reviewing them and testing them on a small sample of dialogues to ensure they are accurate and effective.

Delexicalization to extract templates

One of the common phenomena of extracting the templates is delexicalization. Delexicalization replaces specific words in the training set representing slot values with a generic placeholder token representing the slot. So we created the training dataset by delexicalizing the symptoms. Table 8.1 shows a glimpse of the delexicalized data. After delexicalization, we carefully selected the top most templates and we used these templates for response generation.

8.2.2 Machine Learning-Based NLG.

One of the limitation of the template based NLG is that templates are static and inflexible, meaning that they can't easily be adapted to new situations or changes in the data. So we have considered

Delexicalized Sentence		
<m_test>, <m_test> test accounter;</m_test></m_test>		
<symptom> ສະຜ ຜາລອຍ</symptom>		
<symptom> వెన్తున్నాయా?</symptom>		

Figure 8.1 Delexicalization for NLG data generation.

Machine Learning for response generation. Given training data of N samples containing dialog states and response tuples $D = \{(A_1, x_1), (A_2, x_2), ..., (A_n, x_n), ..., (A_N, x_N)\}$, our goal is to build a statistical model parameterized by θ to characterize $p_{\theta}(x|A)$. To leverage the sequential structure of the response, one may further decompose the joint probability of x using the chain rule as follows

$$p_{\theta}(x|\mathcal{A}) = \prod_{t=1}^{T} p_{\theta}(x_t|(x_1, x_2..., x_{t-1}, \mathcal{A})$$
(8.3)

Learning θ is performed via maximizing the loglikelihood (MLE) of the conditional probabilities in the equation 8.3 over the entire training dataset. It is given as,

$$\mathcal{L}_{\boldsymbol{\theta}}(\mathcal{D}) = \sum_{n=1}^{|\mathcal{D}|} \sum_{t=1}^{T_n} \log p_{\boldsymbol{\theta}} \left(x_t \mid (x_1, x_2 \dots, x_{t-1}, \mathcal{A}_n) \right)$$
(8.4)

Encoder-decoder models generally do the mapping from the dialog state to sentences. We used various encoder-decoder models to implement the statistical NLG module. The SOTA baseline models used for natural language generation are listed here.

8.3 Baseline Models

8.3.1 TemplateNLG

[185] provided a template-based model (named TemplateNLG) for natural language generation. For TemplateNLG, they extracted templates from the training set and manually added some templates for infrequent dialogue acts.

8.3.2 LSTM for language Generation

The paper [156] uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of fixed dimensionality and then another deep LSTM to decode the target sequence from the vector.

8.3.3 Convolutional Seq2seq model

The paper [61] introduced an architecture based entirely on convolutional neural networks. The encoder and decoder models are both CNN blocks with an attention mechanism between them. Compared to recurrent models, computations over all elements can be fully parallelized during training, and optimization is easier since the number of non-linearities is fixed and independent of the input length.

8.3.4 Transformer model

Vanilla Transformer [157] model is based solely on attention mechanisms and introduces Multi-Head Attention. The encoder and decoder are made of multiple layers, with each layer consisting of Multi-Head Attention and Position-wise Feed-forward sub-layers. This model is currently used in many state-of-the-art sequence-to-sequence and transfer learning tasks.

8.3.5 XLM

XLM [94] is pre-trained on a cross-lingual language modeling task. The model is trained to predict masked tokens in a sentence, considering the context of words from different languages. This task encourages the model to learn representations that can transfer across languages. XLM also includes a cross-lingual alignment module, which learns to project the representations of words in different languages into a shared space. It allows for effective cross-lingual transfer, where the model can transfer knowledge from high-resource to low-resource languages. XLM has achieved state-of-the-art results on a range of cross-lingual NLP tasks

8.3.6 mBART

mBART [98] is a multilingual sequence-to-sequence model. It is an encoder-decoder model and can be used for various NLG tasks. mBART is pre-trained on a large corpus of text from 25 languages, including low-resource languages, by training a masked language model (MLM) and a denoising autoencoder (DAE) on parallel data. The MLM predicts missing words in a sentence, while the DAE learns to reconstruct the original sentence from a corrupted version. During pre-training, mBART uses a shared vocabulary across all languages and incorporates language-specific token embeddings to handle language-specific syntax and semantics. It also uses a language identification token to indicate the language of each sentence.

8.3.7 T5

T5 (Text-to-Text Transfer Transformer) [126] is a language model based on the Transformer architecture that uses a text-to-text approach, where the input and output are both text sequences, and the model learns to transform one sequence into another. This approach allows T5 to handle various tasks without requiring separate training. T5 is pre-trained on a large corpus of text data using a denoising autoencoder approach, where the model is trained to reconstruct the original text from a corrupted version of the text. The pre-training process helps T5 learn general language patterns and enables the model to generate high-quality text.

8.3.8 GPT 2

GPT 2(Generative Pre-trained Transformer 2) [125] is a large-scale pre-trained language model. GPT-2 is trained on a large corpus of text data using a self-supervised learning method called unsupervised multitasking. This approach involves training the model to perform multiple language tasks simultaneously, including predicting masked tokens in a sentence, predicting the next word in a sequence, and predicting the likelihood of a given sentence. mGPT-2 [146] (multilingual GPT-2) is a variant of the GPT-2 language model that is pre-trained on text from multiple languages. Like the original GPT-2 model, mGPT-2 is based on the Transformer architecture and uses self-attention mechanisms to capture long-range dependencies in text. mGPT-2 is trained on a large corpus of text from 25 languages [146].

8.4 Experimental setup

We have used the learning rate of around 1e-4, maximum sequence length of 128, dropout = 0.5, batch size = 4, maximum epochs = 10 to run the baseline NLG models.

We used the following evaluation metrics for our NLG model. They are BLEU scores and slot error rate (SER). BLEU score evaluates how natural the generated utterance is when compared to the original sentence. SER measures the exact matching of the slot tokens in the candidate utterances. SER = (p + q)/M. Where M is the total number of slots in the dialog state, and p,q is the number of missing and redundant slots in the given NLG output. The SER takes values from [0,maximum sequence length=128].

Table 8.1 shows the performance of various natural language generation (NLG) models on a task, with two evaluation metrics: BLEU and SER. BLEU is a measure of how similar the generated text is to the reference text, while SER (slot error rate) measures the rate of errors in filling the slots (variables) in the generated text.

The first row of the table shows the performance of a template-based NLG model on the task, with a BLEU score of 0.35 and a SER of 31.6. This model likely uses pre-defined templates with placeholders for variables, which are filled in based on the dialogue context.

Models	BLEU	SER
TemplateNLG [185]	0.35	0.05
Encoder - decoder LSTM [156]	0.36	4.80
CNN seq2seq [61]	0.38	4.71
Transformer Encoder - decoder [157]	0.41	4.28
mBART [98]	0.51	3.39
XLM [94]	0.52	3.72
T5 [126]	0.51	3.29
GPT2 [125]	0.6	2.57

Table 8.1 Performance of NLG module

GPT2 gave a better rendering of the utterance and a reduced ERR rate compared to other NN models. But, the generated utterances suffer from accurate slot rendering. It reveals the complexity of the dataset and the performance of a pre-trained transformer model for NLG when only limited utterances are available for fine-tuning.

But the deep learning-based NLG was giving simple errors, such as the intent slot value pair, i.e., STATEMENT, (AGE, **24**yrs) was transformed as, The age is **25** yrs by the GPT model Even though LLMs are used, deep learning models generate simple errors, and it might impact the system's performance. So we shifted to template-based NLG in the dialog system instead of deep learning models.

8.5 Summary

This chapter talks about the natural language generation unit in our dialog system. The input to our NLG module is the dialog state that contains the intent and the slot-value information given by the dialog manager. The output of the NLG module is the system response in natural language. We used template-based and machine learning-based NLG models to generate the response from dialog states.

Chapter 9

The Code-Mixed Task Oriented Dialog System

In this final chapter, we present the code-mixed task-oriented dialog system. Our dialog system is designed to help users recommend a specialist based on the user's inputs.¹. We will also compare our model with ChatGPT (ChatGPT March 14, 2023 Version)².



Figure 9.1 Delexicalization for NLG data generation.

9.1 A Glimpse about the system

The graphical user interface (GUI) of our dialog system is built using tkinter³. Tkinter is a popular Python library for creating graphical user interfaces. We have used the models that achieved top performance to build the dialog system. The architecture of the dialog system is given in Figure 9.1.

¹This work is supported by "Touchstone" KCIS Enabler Grant Project, IIIT Hyderabad

²https://chat.openai.com/chat

³https://docs.python.org/3/library/tkinter.html

As mentioned before, our dialog system consists of NLU, dialog manager, and NLG modules. The NLU component of your dialog system is responsible for understanding the user's intent and extracting relevant slots from the user's input. We have used the intent classifier that uses contextual signals to predict the intent of the utterance 4.6.1, and the slot tagging module is the chunk-based module that follows 2 phase mechanism to identify the slots in the utterance 4.6.2.

The underlying frame-based dialog manager is responsible for managing the flow of the conversation and keeping track of the current state of the dialog 6.1.

The template-based NLG (Natural Language Generation) 8.1 is responsible for generating responses based on the user's input and the current state of the dialog. The NLG extracted the pre-defined templates from the code-mixed dialog dataset and used them to generate appropriate responses for the given context.

Our dialog system supports both code-mixed Telugu-English interactions and monolingual interactions. It means that users can switch between Telugu and English during the conversation. It is a helpful feature for multilingual users who are more comfortable expressing themselves in one language or the other or need to use specific technical terms or jargon only available in one language.

Our dialog system supports style accommodation. It is another impressive feature of our dialog system. Style accommodation is a technique that helps the dialog system to understand and adapt to the user's communication style. By adapting to the user's style, the system can provide a more personalized and engaging experience, improving user satisfaction and the system's overall effectiveness. For example, if a user switches from code-mixed to Telugu during the conversation, the dialog system can change its response style to match the user's language preference. This feature ensures that the system's responses are more natural and appropriate. The ability to support style accommodation is a significant advantage, as it allows the system to adjust its response style based on the user's input style. This feature makes the conversation more natural and comfortable for the user.

We went beyond the typical dialog system and included the recommendation system in our dialog dataset. The recommendation system analyzes the user's input and suggests additional symptoms that the user may not have mentioned. We have used content-based collaborative filtering to build our recommendation system 6.2.

Detecting hate speech is essential for any conversational system interacting with users in real time. Hate speech is any form of communication that attacks or dehumanizes individuals or groups based on ethnicity, religion, gender, sexual orientation, or other personal characteristics. Our dialog system uses BERT-based hate speech detection models to detect hate speech in multilingual and code-mixed text, allowing it to flag potentially offensive or harmful messages and respond by terminating the conversation.

Finally, the model takes the list of symptoms, medical conditions, demographics, and so forth from the user and suggests a medical specialist 7.1. For example, if the user mentions symptoms related to an opthalmologist, mBERT can identify this and classify the user as wanting to visit an opthalmologist. We have trained a deep-learning model on a dataset of medical utterances, where each utterance is labeled with the corresponding medical specialist. The model can then learn to recognize patterns in the



Figure 9.2 An interaction with the dialog system with focus on code-mixing

input associated with each specialist and make accurate predictions. We put forth a few conversations obtained from the dialog system.



Figure 9.3 An interaction with the dialog system with focus on the English language

Figure 9.2 illustrates a conversation between a user and a chatbot. The bot introduces itself in the first conversation and collects relevant information from the user. It analyzes the user's input using natural



Figure 9.4 An interaction with the dialog system with focus on the Telugu language

language understanding (NLU). The system tracks the state of the dialog system using the frame-based dialog manager and asks for relevant inputs from the user. The system finally recommends a medical specialist for consultation. The system follows style accommodation, responding in the same language as the user, be it code-mixing or Telugu/English. The bot's eighth conversation shows the workings of the recommender system, which suggested symptoms similar to those expressed by the user, helping the

user identify overlooked symptoms. The bot also follows the doctor-patient interaction protocol, asking if the user wants to provide more details and suggesting a specialist.



Figure 9.5 An interaction with the dialog system for hate speech detection

However, the recommender system's performance has a few issues in this conversation. The system recommended a vague symptom, "pain" (in Telugu), and the symptom "cold" was repeated in Telugu and Hindi languages. This situation highlights the limitations of the multilingual dataset and suggests that a larger dataset would help LLMs learn cross-lingual mappings and infer translations more accurately.

Figure 9.3 shows an English-dominated conversation where the bot accommodates the user's language preferences and successfully recommends a specialist. The recommender system works well, but the recommended symptoms are of mixed languages, reflecting the nature of the multilingual dataset. Even though the dataset is code-mixed, and English language is the embedded(second) language in the dataset. Nevertheless, this conversation showcases the chatbot's ability to process monolingual English input and provide relevant recommendations.

Similarly, in Figure 9.4, we can see the system conversing with the user in the Telugu language.



Figure 9.6 An interaction with the dialog system where all information is given at once.

As mentioned before, a dialog system must identify hate speech and take necessary action. Figure 9.5 depicts our system identifying the hate speech in the dialog and stops the conversation ⁴

In the conversation depicted in Figure 9.6, we can see the importance of a frame-based dialog control system. The user inputs their demographics, symptoms, and medical conditions in a single utterance, and the system responds with the next course of action. The system is flexible enough to fill all necessary frames and identify the next set of slots to be provided by the user.

Table 9.1 provides an overview of the conversation's inner workings, which can be useful in understanding how the frame-based dialog control system operates. The table shows the various frames used in the conversation, such as "Age," "gender," "Symptoms," and "Medical Conditions." It also shows the individual slots within each frame, such as "sugar levels are increasing" and "eyesight is disturbed" under "Symptoms," and demographics such as name, age, and gender as independent slots.

⁴The swear words are used only for the system's demonstration

The table also demonstrates how the system updates the dialog state based on the user's input. For example, when the user provides their age, the system fills the "Age" slot and then moves on to the next frame to be filled. Similarly, when the user provides their symptoms, the system fills the corresponding slots under the "Symptoms" frame and determines the appropriate action based on the user's input.

The system then passes the collected information to a recommendation system, which returns a list of symptoms and medical conditions likely to be associated with the user's condition. The system asks the user if they are experiencing any of these symptoms or conditions. The user responds that they have high fluctuations in their sugar levels and their eyesight is disturbed. These symptoms are further identified as slots.

Finally, the system confirms with the user if any other symptoms or conditions have been missed, to which the user responds negatively. The system then identifies the specialist to be consulted based on the collected information. It passes the information to the medical specialist classification module, which predicts that a general physician is an appropriate specialist. The system informs the user that they need to consult a general physician.

This type of dialog control system can be especially useful in healthcare-related conversations, where accurate and comprehensive information is critical. By organizing the conversation into frames and slots, the system can efficiently collect all necessary information and provide appropriate recommendations or actions.

System	namaskAraM, nenu oVka Medical Assistant. nenu mI symptomsni			
	batti mimmalni oVka specialistni meet avvadAniki help ceswAnu (Hi,			
	I'm a medical assistant. I can help you visit an expert, based on your			
	symptoms.)			
User	Hi my name is pravalika, age is 26, female, i have type 1 diabetes,			
	my eye sight is disturbed and my sugar levels are increasing since			
	past 2 days			
Slots	name: Pravalika, age: 24 years, gender:			
identified	female, medical_condition: type 1 diabetes,			
	symptoms: eye sight is disturbed. sugar			
	levels are increasing, time_period: past 2			
	days			
Dialog state	name: Pravalika, age: 24 years, gender:			
	female, medical_condition: type 1 diabetes,			
	symptoms: eye sight is disturbed. sugar			
	levels are increasing, time_period: past 2			
	days			
System	Any medical tests performed before			
User	yes, i took the sugar test			

Slots	m_test: sugar test		
identified			
Dialog state	name: Pravalika, age: 24 years, gender:		
	female, medical_condition: type 1 diabetes,		
	symptoms: eye sight is disturbed. sugar		
	levels are increasing, time_period: past 2		
	days, m_test: sugar test		
System	Did you use any medicine or pills		
User	yes, i take insulin		
Slots	medicine: insulin		
identified			
Dialog state	name: Pravalika, age: 24 years, gender:		
	female, medical_condition: type 1 diabetes,		
	symptoms: eye sight is disturbed. sugar		
	levels are increasing, time_period: past 2		
	days, m_test: sugar test, medicine: insulin		
Recommendation	type 1 diabetes, diabetic, minus power, diabetes, fluctuations, dia-		
system	betesnu, insulin dependent diabetes, sugar test, kallaku minus power,		
	attack sugar		
	anack, sugar		
System	Are you experiencing any symptoms/conditions such as: type 1 di-		
System	Are you experiencing any symptoms/conditions such as: type 1 di- abetes, diabetic, minus power, diabetes, fluctuations, diabetes <i>nu</i> , in-		
System	Are you experiencing any symptoms/conditions such as: type 1 di- abetes, diabetic, minus power, diabetes, fluctuations, diabetes <i>nu</i> , in- sulin dependent diabetes, sugar test, <i>kallaku</i> minus power, attack,		
System	Are you experiencing any symptoms/conditions such as: type 1 di- abetes, diabetic, minus power, diabetes, fluctuations, diabetes <i>nu</i> , in- sulin dependent diabetes, sugar test, <i>kallaku</i> minus power, attack, sugar		
System User	Are you experiencing any symptoms/conditions such as: type 1 di- abetes, diabetic, minus power, diabetes, fluctuations, diabetes <i>nu</i> , in- sulin dependent diabetes, sugar test, <i>kallaku</i> minus power, attack, sugar yes, i have high fluctuatios in my sugar levels and my eye sight is		
System User	Are you experiencing any symptoms/conditions such as: type 1 di- abetes, diabetic, minus power, diabetes, fluctuations, diabetes <i>nu</i> , in- sulin dependent diabetes, sugar test, <i>kallaku</i> minus power, attack, sugar yes, i have high fluctuatios in my sugar levels and my eye sight is disturbed.		
System User Slots	Are you experiencing any symptoms/conditions such as: type 1 di- abetes, diabetic, minus power, diabetes, fluctuations, diabetes <i>nu</i> , in- sulin dependent diabetes, sugar test, <i>kallaku</i> minus power, attack, sugar yes, i have high fluctuatios in my sugar levels and my eye sight is disturbed. symptom: high fluctuations,		
System User Slots identified	Are you experiencing any symptoms/conditions such as: type 1 di- abetes, diabetic, minus power, diabetes, fluctuations, diabetes <i>nu</i> , in- sulin dependent diabetes, sugar test, <i>kallaku</i> minus power, attack, sugar yes, i have high fluctuatios in my sugar levels and my eye sight is disturbed. symptom: high fluctuations, medical_condition: sugar levels, symptom:		
System User Slots identified	Are you experiencing any symptoms/conditions such as: type 1 di- abetes, diabetic, minus power, diabetes, fluctuations, diabetes <i>nu</i> , in- sulin dependent diabetes, sugar test, <i>kallaku</i> minus power, attack, sugar yes, i have high fluctuatios in my sugar levels and my eye sight is disturbed. symptom: high fluctuations, medical_condition: sugar levels, symptom: eye sight is distrubed		
System User Slots identified Dialog state	Are you experiencing any symptoms/conditions such as: type 1 di- abetes, diabetic, minus power, diabetes, fluctuations, diabetes <i>nu</i> , in- sulin dependent diabetes, sugar test, <i>kallaku</i> minus power, attack, sugar yes, i have high fluctuatios in my sugar levels and my eye sight is disturbed. symptom: high fluctuations, medical_condition: sugar levels, symptom: eye sight is distrubed name: Pravalika, age: 24 years, gender:		
System User Slots identified Dialog state	Are you experiencing any symptoms/conditions such as: type 1 di- abetes, diabetic, minus power, diabetes, fluctuations, diabetes <i>nu</i> , in- sulin dependent diabetes, sugar test, <i>kallaku</i> minus power, attack, sugar yes, i have high fluctuatios in my sugar levels and my eye sight is disturbed. symptom: high fluctuations, medical_condition: sugar levels, symptom: eye sight is distrubed name: Pravalika, age: 24 years, gender: female, medical_condition: type 1 diabetes,		
System User Slots identified Dialog state	Are you experiencing any symptoms/conditions such as: type 1 di- abetes, diabetic, minus power, diabetes, fluctuations, diabetes <i>nu</i> , in- sulin dependent diabetes, sugar test, <i>kallaku</i> minus power, attack, sugar yes, i have high fluctuatios in my sugar levels and my eye sight is disturbed. symptom: high fluctuations, medical_condition: sugar levels, symptom: eye sight is distrubed name: Pravalika, age: 24 years, gender: female, medical_condition: type 1 diabetes, sugar levels, symptoms: eye sight is		
System User Slots identified Dialog state	Are you experiencing any symptoms/conditions such as: type 1 di- abetes, diabetic, minus power, diabetes, fluctuations, diabetes <i>nu</i> , in- sulin dependent diabetes, sugar test, <i>kallaku</i> minus power, attack, sugar yes, i have high fluctuatios in my sugar levels and my eye sight is disturbed. symptom: high fluctuations, medical_condition: sugar levels, symptom: eye sight is distrubed name: Pravalika, age: 24 years, gender: female, medical_condition: type 1 diabetes, sugar levels, symptoms: eye sight is disturbed. sugar levels are increasing,		
System User Slots identified Dialog state	Are you experiencing any symptoms/conditions such as: type 1 di- abetes, diabetic, minus power, diabetes, fluctuations, diabetes <i>nu</i> , in- sulin dependent diabetes, sugar test, <i>kallaku</i> minus power, attack, sugar yes, i have high fluctuatios in my sugar levels and my eye sight is disturbed. symptom: high fluctuations, medical_condition: sugar levels, symptom: eye sight is distrubed name: Pravalika, age: 24 years, gender: female, medical_condition: type 1 diabetes, sugar levels, symptoms: eye sight is disturbed. sugar levels are increasing, high fluctuations, eye sight is disturbed,		
System User Slots identified Dialog state	Are you experiencing any symptoms/conditions such as: type 1 di- abetes, diabetic, minus power, diabetes, fluctuations, diabetes <i>nu</i> , in- sulin dependent diabetes, sugar test, <i>kallaku</i> minus power, attack, sugar yes, i have high fluctuatios in my sugar levels and my eye sight is disturbed. symptom: high fluctuations, medical_condition: sugar levels, symptom: eye sight is distrubed name: Pravalika, age: 24 years, gender: female, medical_condition: type 1 diabetes, sugar levels, symptoms: eye sight is disturbed. sugar levels are increasing, high fluctuations, eye sight is disturbed, time_period: past 2 days, m_test: sugar		
System User Slots identified Dialog state	Are you experiencing any symptoms/conditions such as: type 1 di- abetes, diabetic, minus power, diabetes, fluctuations, diabetes <i>nu</i> , in- sulin dependent diabetes, sugar test, <i>kallaku</i> minus power, attack, sugar yes, i have high fluctuatios in my sugar levels and my eye sight is disturbed. symptom: high fluctuations, medical_condition: sugar levels, symptom: eye sight is distrubed name: Pravalika, age: 24 years, gender: female, medical_condition: type 1 diabetes, sugar levels, symptoms: eye sight is disturbed. sugar levels are increasing, high fluctuations, eye sight is disturbed, time_period: past 2 days, m.test: sugar test, medicine: insulin		
System User Slots identified Dialog state System	Are you experiencing any symptoms/conditions such as: type 1 di- abetes, diabetic, minus power, diabetes, fluctuations, diabetes <i>nu</i> , in- sulin dependent diabetes, sugar test, <i>kallaku</i> minus power, attack, sugar yes, i have high fluctuatios in my sugar levels and my eye sight is disturbed. symptom: high fluctuations, medical.condition: sugar levels, symptom: eye sight is distrubed name: Pravalika, age: 24 years, gender: female, medical.condition: type 1 diabetes, sugar levels, symptoms: eye sight is disturbed. sugar levels are increasing, high fluctuations, eye sight is disturbed, time_period: past 2 days, m_test: sugar test, medicine: insulin Please let me know if you have missed any.		

Slots	[]
identified	
Dialog state	name: Pravalika, age: 24 years, gender:
	female, medical_condition: type 1 diabetes,
	sugar levels, symptoms: eye sight is
	disturbed. sugar levels are increasing,
	high fluctuations, eye sight is disturbed,
	time_period: past 2 days, m_test: sugar
	test, medicine: insulin
Medical specialist	<pre>`label': `general_physician', `score':</pre>
classification	0.6102713942527771
System	The doctor to be consulted is "general_physician"

Table 9.1 A detailed working of the dialog system

9.2 Comparing our system with ChatGPT

In this section, we compare the performance of our modular dialog system with ChatGPT. ChatGPT⁵ is a large language model created by OpenAI, based on the GPT-3.5 architecture [19]. The GPT-3.5 architecture is built using a transformer-based neural network that has a deep hierarchy of attention mechanisms. This architecture is a state-of-the-art language model that uses unsupervised learning to pre-train on massive amounts of text data and then fine-tune on specific tasks such as language translation or question-answering. Its purpose is to provide human-like interactions with users through natural language processing.

The largest version of the model, GPT-3, has 175 billion parameters. It was trained on a massive dataset of text from the internet and other sources. The training corpus for GPT-3 consisted of over 45 terabytes of text data, which is equivalent to approximately 570,000 years of spoken language. The large size and diversity of the training corpus for GPT-3 allows the model to capture a wide range of patterns and relationships in natural language, resulting in highly accurate and sophisticated language generation capabilities. The model is contextual and it is trained to reject inappropriate requests. However, it has limited knowledge of the world facts after year 2021.

⁵https://chat.openai.com/chat

User Input (Code-Mixed)	'hi I am manvi, <i>nA</i> baby age <i>vaccesi</i> 26 months, f , <i>pApaku bAgA xaggu</i> ,
	fever uMxi , nenu calpol veSAnu pApaku, goVMwu noVppi iMkA jalubu
	kUdA unnAyi, rashes unnAyi body mIxa aMwA. nenu e doctor ni saM-
	praxiMcAli?'
Translation	hi am manvi, my baby age is 26 months, f, baby has bad cough, fever,
	i gave calpol to baby, sore throat and also cold, there are rashes all over
	body.
ChatGPT Response	hAy mAnvi, mI bAbIki unna jalubu, goVMwu noVppi mariyu rARt un-
	natlayiwe, axi maMci dAktar cUdAli. mIru mI pillavAriki e dAktarnu
	kaMtakt ceyAlo weVlusukovaccu. praXama pAwralo, mIru pEru mariyu
	bAylo sagaM waginaMwa gamaniMcAli. koVnni sAmAnya kuxiMpulu
	mariyu jalubula pE sUcanalu anusariMci, mIru pAwraparulo koVnni
	paxArWAlu ivvadaM valla peVxxa mattigA kuxiMpu waguluwuMxi.
	mIru dAktarku kAl ceswe, pillavAri jalubu mariyu goVMwu noVppini
	peVMcukovadaM pAtiMcukovac
Observations	ChatGPT told to consult a doctor but didn't specify the correct med-
	ical specialist that the patient must consult. The whole interaction is
	monolingual, i.e., no style accommodation, such as interacting with the
	user in the same code-mixed language. The ChatGPT didn't give any
	suggestions asking for the symptoms.
Our Dialog system	Our dialog system accommodated to the user's style. It provided sug-
	gestions based on the user's symptoms so that the user could give in-
	formation about missing symptoms. Our dialog system told the user to
	consult a Paediatrician

Table 9.2 Comparing the performance of chatGPT based on our input given in Figure 9.2

User Input (Telugu)	nA peru lakRmI, 47, swrI, oVka 2 neVlala nuMci nadumu noVppi,
	naduswuMte kAlYlYa noVppulu unnAyi, 4 rojula kiMxa jvaraM kUdA
	vacciMxi. I nadumu noVppi bAgA uMxi. nenu e doctor ni saMpraxiM-
	cAli?
Translation	My name is Lakshmi, 47, female, back pain since 2 months, pain in legs
	while walking, also fever since 4 days. This lower back pain is good.
	Which doctor should I consult?
ChatGPT Response	mIru Arogya lakRaNAlu praxarSiMcinA nijayina vExyudunu saMprax-
	iMcadaM sarikAxu. mI prAraMBa ceVMxina lakRaNAlu pramANiM-
	cadAniki, mIru oVka vExyudunu saMpraxiMci salahA poVMxavaccu.
	koVnni kesullo prAraMBa ceVMxina lakRaNAlu sAXAraNaMgA gAne
	uMtAyi, kAnI anya samasyalu unnavi kUdA uMtAyi. aMxuvalla,
	mIru vExyudunu saMpraxiMci pUrwi parIkRaNanu sUciMcAli mariyu
	parIkRala prakriyalu pUrwi ceyAli. mIru pariSIliMcabadina saMxar-
	BaMlo kAni lakRaNAlanu AXAraMgA mIru nirNayaM ceyavac
Observations	ChatGPT told to consult a doctor but didn't specify the specialization.
	ChatGPT's accommodated to the user's input. It recognized the input
	is in Telugu and responded, respectively. The ChatGPT didn't give any
	suggestions asking for the symptoms.
Our Dialog system	Our system accommodated the user's input and responded respectively
	in the Telugu language. It provided suggestions based on the user's
	symptoms so that the user could give information about missing symp-
	toms. It told the user to consult a Gynaecologist

 Table 9.3 Comparing the performance of chatGPT based on our input given in Figure 9.3

hi its Mohan, 27, m, hair fall and dandruff, i take multivitamins, its been
more than a year, i have lots of hair fall and its increasing day by day.
Which doctor should i visit?
Hello Mohan, Based on your concerns regarding hair fall and dandruff,
you may want to consider seeing a dermatologist. Dermatologists spe-
cialize in the diagnosis and treatment of skin, hair, and nail conditions,
and they can help you determine the underlying cause of your hair fall
and provide treatment options. In addition to seeing a dermatologist, it's
also a good idea to evaluate your diet and lifestyle habits that may con-
tribute to hair fall and dandruff. Stress, lack of sleep, and poor nutrition
can all impact the health of your hair. I hope this information helps, and
I wish you the best of luck in finding a solution to your hair concerns!
ChatGPT told to consult a dermatologist based on the symptoms. It
is expected because, ChatGPT was trained on over a 45 terabytes of
English text, which means that it has a greater familiarity with the struc-
ture and patterns of the English language. And has seen a large medical
dataset when compared to ours for the English dataset. It also gave treat-
ment options for the user to manage their condition. Cons: It didn't give
any recommendations about the symptoms.
Our dialog system accommodated to the user's style and responded to
the input in English even though it has been trained on lots of code-
mixed data. It interacted with the user by providing symptom sugges-
tions. It told the user to consult a Dermatologist.

Table 9.4 Comparing the performance of chatGPT based on our input given in Figure 9.6

Our Dialog System	Chat GPT
Our dialog system offers greater control over the	ChatGPT is a generative language model. It uses a
conversation flow and responses. Each module is	deep neural network to generate text similar to the
specifically designed to handle a particular task,	input text it was trained on. Given a prompt or a
such as natural language understanding or response	sequence of words, ChatGPT can generate a new se-
generation, which can help ensure the conversation	quence of words that are likely to follow the given
stays on track and produces appropriate responses.	prompt based on the patterns it has learned from the
	training data. This makes ChatGPT useful for tasks
	such as text completion, summarization, translation,
	etc.
Our dialog system is built using specific modules	ChatGPT's responses may be less transparent be-
with known functions, so it can be easier to under-	cause a complex neural network generates them.
stand how the system makes decisions and generates	
responses	
It can function with even small data.	It cannot function with small data. That's why it
	could not generate the correct specialist for the given
	Telugu dataset.
Easier to maintain and track the errors.	It is a huge neural network, and it is very difficult
	to correct the error. The model is data-hungry and
	huge amounts of data are needed to correct the error.
Cost-effective and computationally effective	Expensive to deploy and maintain
Easier to integrate new models such as recommen-	Doesn't support recommendation and style accom-
dation systems, hate speech detection, and style ac-	modation
commodation.	
Does support code-mixing	Doesn't support code-mixing
It can be made adaptable to various domains	ChatGPT is a generative model. It can learn from
	any dataset making it more flexible than a system
	that is designed for a specific task.
It is reliable for critical domains such as medical	ChatGPT is generic. It is built on various data
	sources.

Table 9.5 Differences between our system and ChatGPT

Chapter 10

Conclusion and Future Work

10.1 Summary

To facilitate the research and development of real-time multilingual medical dialogue systems that can potentially handle code-mixing and assist in telemedicine, we created a dataset containing codemixed doctor and patient conversations called 'Su-Vaid'. The dataset has 3k dialogs with over 29k utterances and includes ten clinical specializations with diverse patients. The dataset contains 3004 Telugu-English doctor-patient conversations. The data has a code-mixing index of 33.3%. Code-mixed utterances are the most spoken utterances in our dataset, followed by Telugu utterances. Our dataset has Telugu as the matrix language and English as the embedded language. The data is annotated with 19 intents and 34 slots.

We have built a task-oriented dialog system that takes the user's utterances on his medical condition and points him to a specialist. We have followed a modular approach to the task-oriented dialog system by implementing the dialog system with the help of its subcomponents, i.e., NLU, DM, and NLG. To address hate speech and abusive comments during the interaction, we have added a hate speech detection module in the dialog system. We have also included a suggestion system to check if the user overlooks any symptoms. We also analyzed the performance of the dataset on several benchmark NLU, DM, Suggestion system, and NLG models. We have also evaluated the performance of the hate speech detection module on its respective datasets. To the best of our knowledge, it is the first work to explore the real-life doctor-patient interactions in the code-mixed domain for the languages Telugu and Dakhini.

10.2 Future Work

As the data is taken from real-life conversations, we have observed that the patients explicitly mention the symptoms only during the start of the conversation and use anaphors while referring to the symptoms throughout the entire conversation. It introduced lots of anaphors in the data. We believe that additional tagging for anaphora resolution will improve the NLU model's performance, thereby enhancing the performance of the downstream tasks. Similarly, ellipses are another challenging and common phenomenon in conversations. Using advanced NLP techniques, we can infer the missing words from the ellipses, which will help in more understanding of conversations. As mentioned above, Telugu is the dominant language in the dataset. The Telugu language has rich morphology. So any morphological analysis will help in further simplifying the dataset.

Also, during recommendations, there exist different variations of the same symptom. We have to normalize the symptom to deal with such variations. Normalizing a symptom means converting different variations of the same symptom into a standardized format to ensure consistency in the analysis and processing of the data. Normalizing such data will help boost the performance of the modules such as NLU, recommendation systems, and medical specialist classification and further improve the dialog systems' accuracy.

End-to-end dialog systems are popular in the current NLP applications. They only require a single model (mostly a generation model or encoder-decoder model) that interacts with the user to perform a specific task. But the disadvantage of such models is that they require a lot of data. We can increase our dataset by transcribing the LifeLine-Tv9 and Dr. ETV doctor-patient conversations. We can even construct larger datasets by extending the corpus to other languages. The current dialog system focuses only on textual input and output. We can also extend the dialog system for spoken languages by integrating Automatic speech recognition (ASR) and Text-to-speech (TTS) modules, given that these modules can process code-mixed datasets.

Also, When dealing with code-mixing, which refers to mixing two or more languages in a single utterance or conversation, typological differences can play an important role in how code-mixing occurs and is processed. For example, languages with different word orders may require speakers to adjust the word order when mixing the languages in a sentence, or they may lead to word order errors or ambiguous structures. Other typological differences that can affect code-mixing include differences in the grammatical categories and inflectional morphology used in different languages, affecting how words are marked for tense or agreement. Understanding the typological differences between languages can help researchers and language processing models to predict better and explain patterns of code-mixing, as well as to develop strategies for handling code-mixed data in natural language processing and language technology applications.

The tasks such as dealing with the anaphors and ellipses, normalizing the symptoms in the dataset, and understanding the typological differences of the languages while dealing with code-mixing, increasing the dataset for building more robust systems, collecting hate speech datasets on doctor-patient interactions, and using the dataset for any downstream tasks such as medical text summarization are left as future scope.

Appendix A

Appendix

A.1 Intent and Slot Tables

Intent	Definition	%
STATEMENT	Utterances that make assertions or claims about the medical con-	24.9%
	dition.	
INFO_REQUEST	Information Request, Utterances that contains questions.	21.1%
ANSWER	Utterances that respond to INFO_REQUEST action in the an-	18.4%
	tecedent.	
ACK	Acknowledge, Utterances that include understanding during the	17.8%
	conversations.	
MEDICAL_ EX-	Utterances that the doctor provides while explaining about the	5.2%
PLANATION	medical condition.	
ACTION_DIR	Action Directive, Utterances that suggest a course of action in-	5.0%
	future to the listener.	
GREETING	Utterances that indicate the start of the conversation	1.7%
DIAGNOSIS	Utterance where the doctor provides diagnosis to the patient's	1.1%
	medical condition	
REPEAT_	Utterances that repeat or paraphrase what was just said to make	1.0%
REPHRASE	the listener understand.	
COMMIT	Utterances that explicitly commit the speaker to some future	0.9%
	course of action.	
SNU	Signal Not Understood, Utterances that indicate a problem in un-	0.8%
	derstanding the antecedent.	
HOLD	Utterances where the communication is broken and is continued	0.5%
~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	in the succeeding ones.	
CONV_OPEN	<i>Conversation Open</i> , Uttlerances that are conventionally used to	0.3%
	start the conversation.	0.0~
SU	Signal Understood, Utterances that are used as a conventional	0.3%
	phrase to indicate that the communication is being understood	
	by both parties	0.1.07
APPRECIATE	Utterance that indicate applauds from the speaker	0.1%
CORRECT	Utterance that the speaker uses to correct preceding information.	0.1%
CONV_CLOSE	<i>Conversation Close</i> , Utterances that indicate end the conversation	<0.1%
UNINTERPRET	Meaning-full information cannot be interpreted in these utter-	<0.1%
	ances because of exterior noise.	.0.1.0
ABANDONED	Utterances with no response.	<0.1%

Table A.1 Intent and their percentage in our corpus

Slot	Definition	%	Examples
symptom	change in bodily function that is ex- perienced by a patient and is associ- ated with a particular disease	16.94%	fever, cholecystitis, heart attack, <i>xaggu</i>
time_period	duration of a symptom or medical condition	16.75%	sunday, 2 hours, 3 days, <i>saMvastarAlu</i>
body_part	Any part or any organ of human body	8.69%	eyes, nose, mouth, hands, elbow, skin, <i>caMkalo</i>
medical_condition	a medical state of disrepair	7.46%	hypertension, diabetes, bp, sugar, asthma, <i>jalubu</i>
bodily_process	an organic process that takes place in the body	5.48%	sleeping, walking, breathing, <i>nixralo</i>
intensity	The amount or degree of strength	4.00%	high, more, heavy, bAgA, eVkkuva
location	the position of symptom/disease	3.67%	up, down, side, beside, lower, upper, <i>kiMxa</i> , <i>pEna</i>
age	patient's age	3.56%	23, 31, 32, 47, 1
negative	patient expressing negativity for symptom or disease or any bodily process	3.41%	no, <i>lEdu</i>
medicine	drug prescribed or under use by pa- tient	3.25%	cholestron, digestron, dewax
normal	patient expressing that his symptom or medical condition has subsided	2.49%	bAnE undi, waggiMxi
body_substance	the substance of the body	2.41%	AMH, cholesterol, c- peptide
name	patient's name (after pseudonymisa- tion)	2.32%	patient's name
mtest	name of the medical test	2.25%	APTT, CBP, CUE, CT- scan, MRI-scan, ECG
food_substance	any substance that can be metabo- lized by an animal	2.11%	rice, dal, pulses, sweet, water, idli, <i>KicadZI</i>
frequency	the number of occurrences within a given time period	2.02%	2 times, 3 sArlu
treatment	care or medical procedure provided to improve the patient's situation	1.98%	dialysis, lasik

Slot	Definition	%	Examples
mtest_value	value obtained for a medical test	1.59%	13.5 (for hemoglobin), 170/80 (for blood pres- sure)
positive	patient expressing positive answer for symptom or disease or any bod- ily process	1.58%	undi, yes, avnu
medical_professional	One who provides health care treat- ment and advice based on formal training and experience	1.36%	dentist, cardiologist, orthopaedist
person	a sick person	1.17%	patient
familial	speaker's relationship with the pa- tient	1.08%	brother, sister, father, mother, mother-in-law
gender	patient's gender	0.80%	male, female
place	patient's consultation place	0.62%	Hyderabad, Warangal, Vijayawada
color	symptom's attributes	0.61%	red, yellow, white
profession	patient's profession (as profession is linked to some symptoms)	0.42%	software engineer, pan shop,
bm	body mass	0.40%	35 kgs, 3.2kgs
habit	day-to-day habits that deteriorate a person's health	0.35%	drinking, smoking
medicine_dose	the quantity of medicine consumed by the patient	0.32%	250mg, 500mg, 6000IU
marital_status	is the patient married or not	0.27%	married, single
hospital_name	the hospital that the patient has con- sulted or referred	0.21%	Warangal Hospital, Vi- jaya Hospital,
trauma	the patient has been involved in an accident	0.15%	accident, burn
medical_apparatus	a physical injury occurred to the pa- tient	0.11%	laser
organism	foreign body diagnosed in the pa- tient.	0.06%	bacteria, fungus, bug

Table A.2 Slots with their definitions, examples and its percentage in the corpus.

Bibliography

- G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005.
- [2] G. Aguilar, F. AlGhamdi, V. Soto, M. Diab, J. Hirschberg, and T. Solorio. Named entity recognition on code-switched data: Overview of the calcs 2018 shared task. *arXiv preprint arXiv:1906.04138*, 2019.
- [3] G. Aguilar, S. Kar, and T. Solorio. Lince: A centralized benchmark for linguistic code-switching evaluation. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1803–1813, 2020.
- [4] G. Aguilar and T. Solorio. From english to code-switching: Transfer learning with strong morphological clues. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8033–8044, 2020.
- [5] J. Allen. Natural language understanding. Benjamin-Cummings Publishing Co., Inc., 1995.
- [6] E. Alsentzer, J. Murphy, W. Boag, W.-H. Weng, D. Jindi, T. Naumann, and M. McDermott. Publicly available clinical bert embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78, 2019.
- [7] P. Angara, M. Jiménez, K. Agarwal, H. Jain, R. Jain, U. Stege, S. Ganti, H. A. M "uller, and J. W. Ng. Foodie fooderson a conversational agent for the smart kitchen. In *CASCON*, pages 247–253, 2017.
- [8] K. Bali, J. Sharma, M. Choudhury, and Y. Vyas. "i am borrowing ya mixing?" an analysis of english-hindi code mixing in facebook. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 116–126, 2014.
- [9] K. Ball and D. Garrette. Part-of-speech tagging for code-switched, transliterated texts without explicit language identification. In *Proceedings of the 2018 conference on empirical methods in natural language* processing, pages 3084–3089, 2018.
- [10] S. Banerjee, N. Moghe, S. Arora, and M. M. Khapra. A dataset for building code-mixed goal oriented conversation systems. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3766–3780, Santa Fe, New Mexico, USA, Aug. 2018. Association for Computational Linguistics.

- [11] U. Barman, J. Wagner, and J. Foster. Part-of-speech tagging of code-mixed social media content: Pipeline, stacking and joint modelling. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 30–39, Austin, Texas, Nov. 2016. Association for Computational Linguistics.
- [12] A. Bentahila and E. E. Davies. Code-switching and language dominance. In Advances in psychology, volume 83, pages 443–458. Elsevier, 1992.
- [13] A. Bhargava, A. Celikyilmaz, D. Hakkani-Tür, and R. Sarikaya. Easy contextual intent prediction and slot detection. In 2013 ieee international conference on acoustics, speech and signal processing, pages 8337–8341. IEEE, 2013.
- [14] T. W. Bickmore, D. Utami, R. Matsuyama, and M. K. Paasche-Orlow. Improving access to online health information with conversational agents: a randomized controlled experiment. *Journal of medical Internet research*, 18(1):e5239, 2016.
- [15] A. Bohra, D. Vijay, V. Singh, S. S. Akhtar, and M. Shrivastava. A dataset of Hindi-English code-mixed social media text for hate speech detection. In *Proceedings of the Second Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media*, pages 36–41, New Orleans, Louisiana, USA, June 2018. Association for Computational Linguistics.
- [16] A. Bohra, D. Vijay, V. Singh, S. S. Akhtar, and M. Shrivastava. A dataset of hindi-english code-mixed social media text for hate speech detection. In *Proceedings of the second workshop on computational modeling of people's opinions, personality, and emotions in social media*, pages 36–41, 2018.
- [17] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. *arXiv preprint arXiv:1301.7363*, 2013.
- [18] J. B. Brown, M. Stewart, and B. L. Ryan. Outcomes of patient-provider interaction. In *The Routledge Handbook of Health Communication*, pages 155–176. Routledge, 2003.
- T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry,
 A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing* systems, 33:1877–1901, 2020.
- [20] P. Budzianowski, T.-H. Wen, B.-H. Tseng, I. Casanueva, S. Ultes, O. Ramadan, and M. Gasic. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of the* 2018 Conference on Empirical Methods in Natural Language Processing, pages 5016–5026, 2018.
- [21] P. S. Byrne and B. E. Long. Doctors talking to patients: A study of the verbal behavior of general practitioners consulting in their surgeries. 1976.
- [22] B. R. Chakravarthi. Navya jose, shardul suryawanshi, elizabeth sherly, and john philip mc-crae. 2020a. a sentiment analysis dataset for codemixed malayalam-english. In *Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL)*, pages 177–184.
- [23] B. R. Chakravarthi, N. Jose, S. Suryawanshi, E. Sherly, and J. P. McCrae. A sentiment analysis dataset for code-mixed malayalam-english. In *Proceedings of the 1st Joint Workshop on Spoken Language Tech*-

nologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL), pages 177–184, 2020.

- [24] B. R. Chakravarthi, N. Jose, S. Suryawanshi, E. Sherly, and J. P. McCrae. A sentiment analysis dataset for code-mixed Malayalam-English. In *Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL)*, pages 177–184, Marseille, France, May 2020. European Language Resources association.
- [25] B. R. Chakravarthi, P. K. Kumaresan, R. Sakuntharaj, A. K. Madasamy, S. Thavareesan, P. B, S. Chinnaudayar Navaneethakrishnan, J. P. McCrae, and T. Mandl. Overview of the HASOC-DravidianCodeMix Shared Task on Offensive Language Detection in Tamil and Malayalam. In *Working Notes of FIRE 2021* - Forum for Information Retrieval Evaluation. CEUR, 2021.
- [26] B. R. Chakravarthi and V. Muralidaran. Findings of the shared task on hope speech detection for equality, diversity, and inclusion. In *Proceedings of the First Workshop on Language Technology for Equality, Diversity and Inclusion*, pages 61–72, Kyiv, Apr. 2021. Association for Computational Linguistics.
- [27] B. R. Chakravarthi and V. Muralidaran. Findings of the shared task on hope speech detection for equality, diversity, and inclusion. In *Proceedings of the First Workshop on Language Technology for Equality, Diversity and Inclusion*, pages 61–72, 2021.
- [28] B. R. Chakravarthi, V. Muralidaran, R. Priyadharshini, and J. P. McCrae. Corpus creation for sentiment analysis in code-mixed tamil-english text. In *Proceedings of the 1st Joint Workshop on Spoken Lan*guage Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL), pages 202–210, 2020.
- [29] B. R. Chakravarthi, R. Priyadharshini, N. Jose, A. Kumar M, T. Mandl, P. K. Kumaresan, R. Ponnusamy, H. R L, J. P. McCrae, and E. Sherly. Findings of the shared task on offensive language identification in Tamil, Malayalam, and Kannada. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages*, pages 133–145, Kyiv, Apr. 2021. Association for Computational Linguistics.
- [30] B. R. Chakravarthi, R. Priyadharshini, V. Muralidaran, N. Jose, S. Suryawanshi, E. Sherly, and J. P. Mc-Crae. Dravidiancodemix: Sentiment analysis and offensive language identification dataset for dravidian languages in code-mixed text. arXiv preprint arXiv:2106.09460, 2021.
- [31] B. R. Chakravarthi, R. Priyadharshini, V. Muralidaran, S. Suryawanshi, N. Jose, E. Sherly, and J. P. Mc-Crae. Overview of the track on sentiment analysis for dravidian languages in code-mixed text. In *Forum for Information Retrieval Evaluation*, pages 21–24, 2020.
- [32] C. Chelba, M. Mahajan, and A. Acero. Speech utterance classification. In 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03)., volume 1, pages I–I. IEEE, 2003.

- [33] Q. Chen, Z. Zhuo, and W. Wang. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*, 2019.
- [34] S. Chen, G. Aguilar, A. Srinivasan, M. Diab, and T. Solorio. Calcs 2021 shared task: Machine translation for code-switched data. *arXiv preprint arXiv:2202.09625*, 2022.
- [35] Y.-N. Chen, D. Hakkani-Tür, J. Gao, and L. Deng. End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In *Interspeech*, pages 3245–3249, 2016.
- [36] S. Chopra, R. Sawhney, P. Mathur, and R. R. Shah. Hindi-english hate speech detection: Author profiling, debiasing, and practical perspectives. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 386–393, 2020.
- [37] K. M. Colby, S. Weber, and F. D. Hilf. Artificial paranoia. Artificial Intelligence, 2(1):1–25, 1971.
- [38] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv*:1911.02116, 2019.
- [39] A. Conneau, G. Lample, M. Ranzato, L. Denoyer, and H. Jégou. Word translation without parallel data. arXiv preprint arXiv:1710.04087, 2017.
- [40] M. G. Core and J. Allen. Coding dialogs with the damsl annotation scheme. In AAAI fall symposium on communicative action in humans and machines, volume 56, pages 28–35. Boston, MA, 1997.
- [41] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.
- [42] R. A. Cote. Architecture of snomed: its contribution to medical language processing. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, page 74. American Medical Informatics Association, 1986.
- [43] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, T. Lavril, et al. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. arXiv preprint arXiv:1805.10190, 2018.
- [44] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9268–9277, 2019.
- [45] K. Davey, V. Ravishankar, N. Mehta, T. Ahluwalia, J. Blanchard, J. Smith, and K. Douglass. A qualitative study of workplace violence among healthcare providers in emergency departments in india. *International journal of emergency medicine*, 13(1):1–9, 2020.
- [46] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.
- [47] A. S. Doğruöz, S. Sitaram, B. E. Bullock, and A. J. Toribio. A survey of code-switching: Linguistic and social perspectives for language technologies. In *The Joint Conference of the 59th Annual Meeting*

of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021). Association for Computational Linguistics, 2021.

- [48] C. Dong, Y. Li, H. Gong, M. Chen, J. Li, Y. Shen, and M. Yang. A survey of natural language generation. arXiv preprint arXiv:2112.11739, 2021.
- [49] S. Dowlagar and R. Mamidi. Cmsaone@ dravidian-codemix-fire2020: A meta embedding and transformer model for code-mixed sentiment analysis on social media text. 2020.
- [50] S. Dowlagar and R. Mamidi. Gated convolutional sequence to sequence based learning for english-hingilsh code-switched machine translation. In *Proceedings of the Fifth Workshop on Computational Approaches* to Linguistic Code-Switching, pages 26–30, 2021.
- [51] S. Dowlagar and R. Mamidi. Graph convolutional networks with multi-headed attention for code-mixed sentiment analysis. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages*, pages 65–72, 2021.
- [52] S. Dowlagar and R. Mamidi. Hasocone@ fire-hasoc2020: Using bert and multilingual bert models for hate speech detection. arXiv preprint arXiv:2101.09007, 2021.
- [53] S. Dowlagar and R. Mamidi. Offlangone@ dravidianlangtech-eacl2021: Transformers with the class balanced loss for offensive language identification in dravidian code-mixed text. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages*, pages 154–159, 2021.
- [54] S. Dowlagar and R. Mamidi. A pre-trained transformer and cnn model with joint language id and part-ofspeech tagging for code-mixed social-media text. pages 367–374, 2021.
- [55] S. Dowlagar and R. Mamidi. A survey of recent neural network models on code-mixed indian hate speech data. In *Forum for Information Retrieval Evaluation*, pages 67–74, 2021.
- [56] S. Dowlagar and R. Mamidi. Hate speech detection on code-mixed dataset using a fusion of custom and pre-trained models with profanity vector augmentation. SN Computer Science, 3(4):1–17, 2022.
- [57] R. FB. Medical subject headings. Bulletin of the Medical Library Association, 51:114–116, 1963.
- [58] G. Ferguson. Classroom code-switching in post-colonial contexts: Functions, attitudes and policies. AILA review, 16(1):38–51, 2003.
- [59] J. L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378– 382, 1971.
- [60] B. Gambäck and A. Das. On measuring the complexity of code-mixing. In Proceedings of the 11th International Conference on Natural Language Processing, Goa, India, pages 1–7, 2014.
- [61] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252, 2017.
- [62] D. Goddeau, H. Meng, J. Polifroni, S. Seneff, and S. Busayapongchai. A form-based dialogue manager for spoken language applications. In *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*, volume 2, pages 701–704. IEEE, 1996.

- [63] C.-W. Goo, G. Gao, Y.-K. Hsu, C.-L. Huo, T.-C. Chen, K.-W. Hsu, and Y.-N. Chen. 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, 2018.
- [64] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov. Learning word vectors for 157 languages. In Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018), 2018.
- [65] J. Gumperz. John (1964). hindi-punjabi code-switching in delhi. In Proceedings of the Ninth International Congress of Linguistics. Mouton: The Hague.
- [66] J. J. Gumperz. The sociolinguistic significance of conversational code-switching. *RELC journal*, 8(2):1–34, 1977.
- [67] A. Gupta, P. Zhang, G. Lalwani, and M. Diab. Casa-nlu: Context-aware self-attentive natural language understanding for task-oriented chatbots. 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 1285–1290, 2019.
- [68] D. Gupta, P. Lenka, A. Ekbal, and P. Bhattacharyya. Uncovering code-mixed challenges: A framework for linguistically driven question generation and neural based question answering. In *Proceedings of the* 22nd Conference on Computational Natural Language Learning, pages 119–130, Brussels, Belgium, Oct. 2018. Association for Computational Linguistics.
- [69] P. Haffner, G. Tur, and J. H. Wright. Optimizing svms for complex call classification. In 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03)., volume 1, pages I–I. IEEE, 2003.
- [70] A. Hande, R. Priyadharshini, and B. R. Chakravarthi. KanCMD: Kannada CodeMixed dataset for sentiment analysis and offensive language detection. In *Proceedings of the Third Workshop on Computational Modeling of People's Opinions, Personality, and Emotion's in Social Media*, pages 54–63, Barcelona, Spain (Online), Dec. 2020. Association for Computational Linguistics.
- [71] A. Hande, R. Priyadharshini, and B. R. Chakravarthi. Kancmd: Kannada codemixed dataset for sentiment analysis and offensive language detection. In *Proceedings of the Third Workshop on Computational Modeling of People's Opinions, Personality, and Emotion's in Social Media*, pages 54–63, 2020.
- [72] C. T. Hemphill, J. J. Godfrey, and G. R. Doddington. 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, 1990.
- [73] M. Henderson, B. Thomson, and J. D. Williams. 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, 2014.
- [74] R. R. Heredia and J. Altarriba. Bilingual language mixing: Why do bilinguals code-switch? Current Directions in Psychological Science, 10(5):164–168, 2001.

- [75] J. Heritage and D. W. Maynard. Problems and prospects in the study of physician-patient interaction: 30 years of research. Annu. Rev. Sociol., 32:351–374, 2006.
- [76] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.
- [77] J. Holmes and N. Wilson. An introduction to sociolinguistics. Routledge, 2017.
- [78] J. Hoxha, P. Chandar, Z. He, J. Cimino, D. Hanauer, and C. Weng. Dream: Classification scheme for dialog acts in clinical research query mediation. *Journal of biomedical informatics*, 59:89–101, 2016.
- [79] Z. Huang, W. Xu, and K. Yu. Bidirectional lstm networks for improved sentence classification. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1565–1573, 2015.
- [80] M. I. Jordan. Serial order: A parallel distributed processing approach. In Advances in psychology, volume 121, pages 471–495. Elsevier, 1997.
- [81] A. Joshi. Processing of sentences with intra-sentential code-switching. In *Coling 1982: Proceedings of the Ninth International Conference on Computational Linguistics*, 1982.
- [82] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759, 2016.
- [83] D. Jurafsky. Speech & language processing. Pearson Education India, 2000.
- [84] B. B. Kachru. Toward structuring code-mixing: an indian perspective. 1978.
- [85] D. Kakwani, A. Kunchukuttan, S. Golla, N. Gokul, A. Bhattacharyya, M. M. Khapra, and P. Kumar. inlpsuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for indian languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4948–4961, 2020.
- [86] T. Kanakagiri and K. Radhakrishnan. Task-oriented dialog systems for dravidian languages. In Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages, pages 85–93, 2021.
- [87] K. Kedia and A. Nandy. indicnlp@ kgp at dravidianlangtech-eacl2021: Offensive language identification in dravidian languages. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages*, pages 330–335, 2021.
- [88] S. Khairat, S. Liu, T. Zaman, B. Edson, R. Gianforcaro, and R. Kukafka. Senior medication assistant: Design and development of a conversational system. In *AMIA Annual Symposium Proceedings*, volume 2018, pages 682–691. American Medical Informatics Association, 2018.
- [89] S. Khanuja, S. Dandapat, A. Srinivasan, S. Sitaram, and M. Choudhury. Gluecos: An evaluation benchmark for code-switched nlp. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3575–3585, 2020.
- [90] Y. Kim. 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, Oct. 2014. Association for Computational Linguistics.

- [91] Y. Kim. 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, Oct. 2014. Association for Computational Linguistics.
- [92] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv* preprint arXiv:1609.02907, 2016.
- [93] R. Kumar, A. N. Reganti, A. Bhatia, and T. Maheshwari. Aggression-annotated corpus of hindi-english code-mixed data. *arXiv preprint arXiv:1803.09402*, 2018.
- [94] G. Lample and A. Conneau. Cross-lingual language model pretraining. *Transactions of the Association for Computational Linguistics*, 7:597–610, 2019.
- [95] G. Lample, A. Conneau, L. Denoyer, and M. Ranzato. Unsupervised machine translation using monolingual corpora only. arXiv preprint arXiv:1711.00043, 2017.
- [96] K. Lang. Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings 1995*, pages 331– 339. Elsevier, 1995.
- [97] L. Laranjo, A. G. Dunn, H. L. Tong, A. B. Kocaballi, J. Chen, R. Bashir, D. Surian, B. Gallego, F. Magrabi, A. Y. Lau, et al. Conversational agents in healthcare: a systematic review. *Journal of the American Medical Informatics Association*, 25(9):1248–1258, 2018.
- [98] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461, 2019.
- [99] X. Li, Y.-N. Chen, L. Li, J. Gao, and A. Celikyilmaz. End-to-end task-completion neural dialogue systems. In Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 733–743, 2017.
- [100] B. Liu and I. Lane. Attention-based recurrent neural network models for joint intent detection and slot filling. *Interspeech 2016*, pages 685–689, 2016.
- [101] S. Louvan and B. Magnini. Recent neural methods on slot filling and intent classification for task-oriented dialogue systems: A survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 480–496, Barcelona, Spain (Online), Dec. 2020. International Committee on Computational Linguistics.
- [102] X. Lu, Y. Zhang, J. Li, and S. Zong. Doctor recommendation in online health forums via expertise learning. arXiv preprint arXiv:2203.02932, 2022.
- [103] T. C. Luo, A. Aguilera, C. R. Lyles, and C. A. Figueroa. Promoting physical activity through conversational agents: Mixed methods systematic review. *Journal of Medical Internet Research*, 23(9):e25486, 2021.
- [104] M. Mager, Ö. Çetinoğlu, and K. Kann. Subword-level language identification for intra-word codeswitching. 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 2005–2011, 2019.
- [105] S. K. Majumder, T. Mondal, M. J. Deen, R. K. Ward, G. Sharma, T. Rahman, L. Fournier, D. Balaguru, and A. Shridhar. Real-time mobile-based parkinson's disease detection using voice data through a telemedicine approach. In 2018 IEEE International Conference on Healthcare Informatics (ICHI), pages 274–275. IEEE, 2018.
- [106] T. Mandl, S. Modha, P. Majumder, D. Patel, M. Dave, C. Mandlia, and A. Patel. 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, 2019.
- [107] C. Mao, Q. Zhu, R. Chen, and W. Su. Automatic medical specialty classification based on patients' description of their symptoms. *BMC Medical Informatics and Decision Making*, 23(1):1–9, 2023.
- [108] P. Mathur, R. Shah, R. Sawhney, and D. Mahata. Detecting offensive tweets in hindi-english code-switched language. In *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*, pages 18–26, 2018.
- [109] D. Mave, S. Maharjan, and T. Solorio. Language identification and analysis of code-switched social media text. In *Proceedings of the third workshop on computational approaches to linguistic code-switching*, pages 51–61, 2018.
- [110] M. F. McTear. Modelling spoken dialogues with state transition diagrams: experiences with the cslu toolkit. *development*, 5(7), 1998.
- [111] G. Mesnil, X. He, L. Deng, and Y. Bengio. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Interspeech*, pages 3771–3775, 2013.
- [112] A. Mohan, D. Chakraborty, A. Gupta, A. Wadhwa, and P. Kumaraguru. Diabetes assistant: A conversational system for diabetes management. In *Proceedings of the 4th Workshop on HealthNLP@ACL*, pages 80–85, 2020.
- [113] N. Mrkšić, D. O. Séaghdha, T.-H. Wen, B. Thomson, and S. Young. Neural belief tracker: Data-driven dialogue state tracking. arXiv preprint arXiv:1606.03777, 2016.
- [114] C. Myers-Scotton. Common and uncommon ground: Social and structural factors in codeswitching. *Language in society*, 22(4):475–503, 1993.
- [115] C. Myers-Scotton. Duelling languages: Grammatical structure in codeswitching. Oxford University Press, 1997.
- [116] N. Nayak, D. Hakkani-Tür, M. A. Walker, and L. P. Heck. To plan or not to plan? discourse planning in slot-value informed sequence to sequence models for language generation. In *INTERSPEECH*, pages 3339–3343, 2017.
- [117] J. Ni, Y. Cai, G. Tang, and Y. Xie. Collaborative filtering recommendation algorithm based on tf-idf and user characteristics. *Applied Sciences*, 11(20):9554, 2021.
- [118] M.-H. Park, J.-H. Hong, and S.-B. Cho. Location-based recommendation system using bayesian user's preference model in mobile devices. In *International conference on ubiquitous intelligence and computing*, pages 1130–1139. Springer, 2007.

- [119] J. Pfeiffer, A. Kamath, A. Rücklé, K. Cho, and I. Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, 2021.
- [120] J. Pfeiffer, A. Rücklé, C. Poth, A. Kamath, I. Vulić, S. Ruder, K. Cho, and I. Gurevych. Adapterhub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, 2020.
- [121] T. Pires, E. Schlinger, and D. Garrette. How multilingual is multilingual bert? *arXiv preprint arXiv:1906.01502*, 2019.
- [122] F. Poletto, V. Basile, M. Sanguinetti, C. Bosco, and V. Patti. Resources and benchmark corpora for hate speech detection: a systematic review. *Language Resources and Evaluation*, 55(2):477–523, 2021.
- [123] S. Poplack. Syntactic structure and social function of code-switching, volume 2. Centro de Estudios Puertorriqueños, [City University of New York], 1978.
- [124] R. Priyadharshini, B. R. Chakravarthi, M. Vegupatti, and J. P. McCrae. Named entity recognition for code-mixed indian corpus using meta embedding. In 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), pages 68–72. IEEE, 2020.
- [125] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1:9, 2019.
- [126] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.
- [127] L. A. Ramshaw and M. P. Marcus. Text chunking using transformation-based learning. In Natural language processing using very large corpora, pages 157–176. Springer, 1999.
- [128] P. Rani, S. Suryawanshi, K. Goswami, B. R. Chakravarthi, T. Fransen, and J. P. McCrae. A comparative study of different state-of-the-art hate speech detection methods in hindi-english code-mixed data. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 42–48, 2020.
- [129] C. Raymond and G. Riccardi. Generative and discriminative algorithms for spoken language understanding. In *Interspeech 2007-8th Annual Conference of the International Speech Communication Association*, 2007.
- [130] N. M. Rémy, T. T. Martial, and T. D. Clémentin. The prediction of good physicians for prospective diagnosis using data mining. *Informatics in medicine unlocked*, 12:120–127, 2018.
- [131] F. Ricci, L. Rokach, and B. Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.
- [132] J. D. Robinson. An interactional structure of medical activities during acute visits and its implications for patients' participation. *Health Communication*, 15(1):27–59, 2003.
- [133] R. Rojowiec, B. Roth, and M. Fink. Intent recognition in doctor-patient interviews. In Proceedings of The 12th Language Resources and Evaluation Conference, pages 702–709, 2020.

- [134] S. Ruder, I. Vulić, and A. Søgaard. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*, 65:569–631, 2019.
- [135] H. Sacks, E. A. Schegloff, and G. Jefferson. A simplest systematics for the organization of turn taking for conversation. In *Studies in the organization of conversational interaction*, pages 7–55. Elsevier, 1978.
- [136] D. Saha, N. Paharia, D. Chakraborty, P. Saha, and A. Mukherjee. Hate-alert@ dravidianlangtecheacl2021: Ensembling strategies for transformer-based offensive language detection. arXiv preprint arXiv:2102.10084, 2021.
- [137] K. R. Sane, S. Kolla, S. R. Sane, V. K. Srirangam, and R. Mamidi. Corpus and baseline system for hate speech detection in telugu-english code-mixed tweets.
- [138] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, Minnesota Univ Minneapolis Dept of Computer Science, 2000.
- [139] E. A. Schegloff, G. Jefferson, and H. Sacks. The preference for self-correction in the organization of repair in conversation. *Language*, 53(2):361–382, 1977.
- [140] A. Schmidt and M. Wiegand. A survey on hate speech detection using natural language processing. In Proceedings of the fifth international workshop on natural language processing for social media, pages 1–10, 2017.
- [141] A. Schmidt and M. Wiegand. A survey on hate speech detection using natural language processing. In Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media, April 3, 2017, Valencia, Spain, pages 1–10. Association for Computational Linguistics, 2019.
- [142] S. Schuster, S. Gupta, R. Shah, and M. Lewis. Cross-lingual transfer learning for multilingual task oriented dialog. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 3795–3805, 2019.
- [143] C. M. Scotton and W. Ury. Bilingual strategies: The social functions of code-switching. 1977.
- [144] T. Shen, T. Zhou, G. Long, J. Jiang, S. Pan, and C. Zhang. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.
- [145] S. Shivaprasad and M. Sadanandam. Identification of regional dialects of telugu language using text independent speech processing models. *International Journal of Speech Technology*, pages 1–8, 2020.
- [146] O. Shliazhko, A. Fenogenova, M. Tikhonova, V. Mikhailov, A. Kozlova, and T. Shavrina. mgpt: Few-shot learners go multilingual. *arXiv preprint arXiv:2204.07580*, 2022.
- [147] S. Sitaram, K. R. Chandu, S. K. Rallabandi, and A. W. Black. A survey of code-switched speech and language processing. arXiv preprint arXiv:1904.00784, 2019.
- [148] T. Solorio and Y. Liu. Learning to predict code-switching points. In Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, pages 973–981, Honolulu, Hawaii, Oct. 2008. Association for Computational Linguistics.

- [149] D. Sravani, L. Kameswari, and R. Mamidi. Political discourse analysis: A case study of code mixing and code switching in political speeches. In *Proceedings of the Fifth Workshop on Computational Approaches* to Linguistic Code-Switching, pages 1–5, 2021.
- [150] V. Srivastava and M. Singh. Phinc: A parallel hinglish social media code-mixed corpus for machine translation. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 41–49, 2020.
- [151] M. A. Stewart. Effective physician-patient communication and health outcomes: a review. CMAJ: Canadian medical association journal, 152(9):1423–33, 1995.
- [152] T. Stivers and A. Majid. Domains of knowledge and responsibility: Questioning in acute pediatric encounters. In *the 14th World Congress of Applied Linguistics (AILA)*, 2005.
- [153] A. Stolcke, K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, P. Taylor, R. Martin, C. V. Ess-Dykema, and M. Meteer. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3):339–373, 2000.
- [154] P. M. Strong and R. Dingwall. *The ceremonial order of the clinic: Parents, doctors and medical bureaucracies.* Routledge, 2018.
- [155] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019.
- [156] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. Advances in neural information processing systems, 27, 2014.
- [157] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
- [158] C. H. Vickers, R. Goble, and S. K. Deckert. Third party interaction in the medical context: Code-switching and control. *Journal of Pragmatics*, 84:154–171, 2015.
- [159] Y. Vyas, S. Gella, J. Sharma, K. Bali, and M. Choudhury. Pos tagging of english-hindi code-mixed social media content. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 974–979, 2014.
- [160] Y.-Y. Wang, L. Deng, and A. Acero. Spoken language understanding. *IEEE Signal Processing Magazine*, 22(5):16–31, 2005.
- [161] Z. Wei, Q. Liu, B. Peng, H. Tou, T. Chen, X. Huang, K.-f. Wong, and X. Dai. Task-oriented dialogue system for automatic diagnosis. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–207, Melbourne, Australia, July 2018. Association for Computational Linguistics.
- [162] J. Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.

- [163] H. Weld, X. Huang, S. Long, J. Poon, and S. C. Han. A survey of joint intent detection and slot-filling models in natural language understanding. *arXiv preprint arXiv:2101.08091*, 2021.
- [164] T.-H. Wen, M. Gasic, D. Kim, N. Mrksic, P.-H. Su, D. Vandyke, and S. Young. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. *arXiv preprint arXiv:1508.01755*, 2015.
- [165] T.-H. Wen, M. Gasic, N. Mrksic, P.-H. Su, D. Vandyke, and S. Young. Semantically conditioned lstmbased natural language generation for spoken dialogue systems. arXiv preprint arXiv:1508.01745, 2015.
- [166] T.-H. Wen, D. Vandyke, N. Mrkšić, M. Gasic, L. M. R. Barahona, P.-H. Su, S. Ultes, and S. Young. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference* of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, pages 438–449, 2017.
- [167] C. West. Routine complications: Troubles with talk between doctors and patients. 1984.
- [168] G. I. Winata, Z. Lin, and P. Fung. Learning multilingual meta-embeddings for code-switching named entity recognition. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 181–186, Florence, Italy, Aug. 2019. Association for Computational Linguistics.
- [169] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771, 2019.
- [170] N. I. Wood. Departing from doctor-speak: A perspective on code-switching in the medical setting. *Journal of general internal medicine*, 34(3):464–466, 2019.
- [171] C.-S. Wu, S. Hoi, R. Socher, and C. Xiong. Tod-bert: Pre-trained natural language understanding for task-oriented dialogue. arXiv preprint arXiv:2004.06871, 2020.
- [172] D. Wu, L. Ding, F. Lu, and J. Xie. Slotrefine: A fast non-autoregressive model for joint intent detection and slot filling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing* (EMNLP), pages 1932–1937, 2020.
- [173] S. Wu and M. Dredze. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 833–844, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.
- [174] P. Xu and R. Sarikaya. 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, 2013.
- [175] W. Xu, B. Haider, and S. Mansour. End-to-end slot alignment and recognition for cross-lingual nlu, 2020.
- [176] Z. Yan, N. Duan, P. Chen, M. Zhou, J. Zhou, and Z. Li. Building task-oriented dialogue systems for online shopping. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

- [177] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu. Recurrent neural networks for language understanding. In *Interspeech*, pages 2524–2528, 2013.
- [178] L. Yao, C. Mao, and Y. Luo. Graph convolutional networks for text classification. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 7370–7377, 2019.
- [179] Z. Yirmibeşoğlu and G. Eryiğit. Detecting code-switching between Turkish-English language pair. In Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text, pages 110–115, Brussels, Belgium, Nov. 2018. Association for Computational Linguistics.
- [180] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar. Predicting the type and target of offensive posts in social media. *arXiv preprint arXiv:1902.09666*, 2019.
- [181] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). arXiv preprint arXiv:1903.08983, 2019.
- [182] G. Zeng, W. Yang, Z. Ju, Y. Yang, S. Wang, R. Zhang, M. Zhou, J. Zeng, X. Dong, R. Zhang, et al. Meddialog: A large-scale medical dialogue dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9241–9250, 2020.
- [183] S. Zhang, L. Yao, A. Sun, and Y. Tay. v. ACM Computing Surveys (CSUR), 52(1):1–38, 2019.
- [184] Z. Zhang, R. Takanobu, Q. Zhu, M. Huang, and X. Zhu. Recent advances and challenges in task-oriented dialog systems. *Science China Technological Sciences*, pages 1–17, 2020.
- [185] Q. Zhu, K. Huang, Z. Zhang, X. Zhu, and M. Huang. Crosswoz: A large-scale chinese cross-domain task-oriented dialogue dataset. *Transactions of the Association for Computational Linguistics*, 8:281–295, 2020.