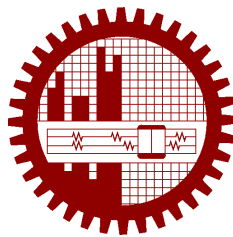


M.Sc. Engg. (CSE) Thesis

Bangla Voice Command Recognition With Context Specific Optimization

Submitted by
Nafis Sadeq
1018052033

Supervised by
Dr. Muhammad Abdullah Adnan



Submitted to
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka, Bangladesh

in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science and Engineering

February 2021

Candidate's Declaration

I, do, hereby, certify that the work presented in this thesis, titled, “Bangla Voice Command Recognition With Context Specific Optimization”, is the outcome of the investigation and research carried out by me under the supervision of Dr. Muhammad Abdullah Adnan, Associate Professor, Department of CSE, BUET.

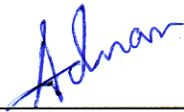
I also declare that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.


Nafis Sadeq

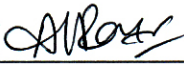
1018052033


The thesis titled “**Bangla Voice Command Recognition With Context Specific Optimization**”, submitted by Nafis Sadeq, Student ID 1018052033, Session October 2018, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfilment of the requirements for the degree of Master of Science in Computer Science and Engineering and approved as to its style and contents on February 27, 2021.

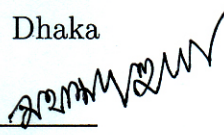
Board of Examiners

1. 

Dr. Muhammad Abdullah Adnan
Associate Professor
Department of CSE, BUET, Dhaka
Chairman
(Supervisor)
2. 

Dr. A.K.M. Ashikur Rahman
Professor and Head
Department of CSE, BUET, Dhaka
Member
(Ex-Officio)
3. 

Dr. A.B.M. Alim Al Islam
Professor
Department of CSE, BUET, Dhaka
Member
4. 

Dr. Rifat Shahriyar
Associate Professor
Department of CSE, BUET, Dhaka
Member
5. 

Dr. Mohammad Nurul Huda
Professor
Department of Computer Science and Engineering
United International University, Dhaka
Member
(External)

Acknowledgement

Firstly, all praise to almighty Allah for assisting me throughout everything.

I would like to express my sincere gratitude to my advisor Dr. Muhammad Abdullah Adnan for the continuous support of my study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my M.Sc. study.

I thank my fellow lab members, specially Sudipta Saha and Shafayat Ahmed, for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last two years. I would also like to thank Samsung R&D Institute Bangladesh for supporting our project with research grant as well as laboratory equipment.

I thank Dr. A.B.M. Alim Al Islam, Dr. Rifat Shahriyar, Dr. Mohammad Nurul Huda and Dr. A.K.M. Ashikur Rahman for their constructive feedback for improving my thesis work further.

Last but not the least, I would like to thank my family: my parents, my wife, and my sister for supporting me throughout the degree program and my life in general.

Dhaka
February 27, 2021

Nafis Sadeq
1018052033

Contents

Candidate’s Declaration	i
Board of Examiners	ii
Acknowledgement	iii
List of Figures	viii
List of Tables	x
List of Algorithms	xii
Abstract	xiii
1 Introduction	1
1.1 Research Objective	1
1.2 Motivation	1
1.3 Present State	1
1.4 Problem Definition	2
1.5 Outline of Methodology	2
1.5.1 Corpus Resource Development	2
1.5.2 Speech Recognition Model Development	4
1.5.3 Context Specific Optimization	4
1.5.4 Other Improvements	4
1.6 Our Contribution	5
2 Literature Review	7
2.1 G2P Conversion	7
2.2 Speech Corpus Preparation	8
2.3 Context-specific Optimization	9
2.4 Semi-supervised ASR Training	10
2.5 Summary of Gap Analysis	12
3 Linguistic Resource Preparation	13

3.1	Phoneme List	13
3.2	Text Corpus and Word Dictionary Preparation	14
3.2.1	Web Crawling	14
3.2.2	Text Cleaning	15
3.2.3	Text Normalization	15
3.2.4	Domain Specific Text Corpus	16
3.2.5	Word Dictionary	16
3.3	G2P Training and Lexicon Preparation	17
3.3.1	Previous Works on G2P	18
3.3.2	Non-Trivial Cases for Transcription	18
3.3.3	Developing an Improved G2P System for Bangla Language	21
3.3.4	G2P Models	24
3.3.5	Experimental Results	24
4	Speech Corpus Preparation	30
4.1	Voice Command Domain Study	30
4.2	Previously Available Speech Corpus	31
4.3	Supervised Speech Corpus Development	31
4.3.1	Data Collection App	32
4.3.2	Summary of Voice Command Specific Corpus	34
4.4	Corpus Generation Using Automated Transcription	34
4.4.1	Background Noise Removal	36
4.4.2	Speaker Diarization	37
4.4.3	Gender Detection	37
4.4.4	Silence Based Segmentation	37
4.4.5	Automatic Transcription Generation	38
4.4.6	Evaluation of Automatic Transcription	39
4.4.7	Summary of Transcribed Corpus	42
4.5	Synthetic Speech Generation for OOV words	42
4.5.1	Out-of-Vocabulary Word List	42
4.5.2	TTS Model	43
4.5.3	Speech Synthesis	43
4.6	Speech Corpus Summary	44
5	Speech Recognition Architecture	45
5.1	Traditional ASR System	45
5.1.1	Speech Feature Extraction	45
5.1.2	Acoustic Model	46
5.1.3	Language Model	46

5.1.4	Phonetic Dictionary	46
5.1.5	Training Phase	47
5.1.6	Decoding Phase	47
5.2	End-to-End System	48
5.2.1	Speech Feature	48
5.2.2	CTC-Attention	49
5.2.3	Language Model	49
5.2.4	Beam Search	51
6	Context Specific Optimization of Voice Commands	52
6.1	Related Works	53
6.2	Our System	54
6.2.1	System Overview	54
6.2.2	End-to-End Architecture	54
6.2.3	Language Model	55
6.2.4	Beam Search	55
6.2.5	Contextual Rescoring	55
6.3	Contextual Corpus Management	57
6.3.1	Contextual Corpus Generation	57
6.3.2	On-device Model Training	57
6.4	Dataset	59
6.4.1	Text Corpus	59
6.4.2	Speech Corpus	59
6.5	Experiments	60
6.5.1	Training Details	60
6.5.2	Test Set	60
6.5.3	Results	60
7	Semi-supervised Speech Recognition	63
7.1	Related Works	64
7.2	Our System	65
7.2.1	Baseline System	65
7.2.2	Semi-supervised System	66
7.3	The Inter-Domain Loss	68
7.3.1	Encoding Procedure	68
7.3.2	Maximum Mean Discrepancy Loss	69
7.3.3	Global Encoding Distance (GED) Loss	70
7.4	Corpus Description	71
7.4.1	Paired Speech Corpus	72

7.4.2	Unpaired Audio Data	72
7.4.3	Unpaired Text Data	72
7.5	Evaluations	73
7.5.1	Test Set	73
7.5.2	Training Details	73
7.5.3	Performance Comparison with External Language Model	73
7.5.4	Performance Comparison of Inter-domain Loss	76
7.5.5	Effect of CTC Weight	76
7.5.6	Effect of Speech Text Ratio	76
7.5.7	Effect of Supervised Loss Ratio	77
7.5.8	Effect of Batch Size	77
8	Evaluation of ASR System	79
8.1	Experiments on Open Domain ASR Task	79
8.1.1	Dataset	79
8.1.2	Training Details	79
8.1.3	Test Set	80
8.1.4	Results	80
8.2	Experiments on Voice Command Task	81
8.2.1	Dataset	81
8.2.2	Training Details	82
8.2.3	Test Set	82
8.2.4	Results	82
9	Conclusion and Future Work	85
9.1	Summary of Research Work	85
9.2	Research Outcome	85
9.3	Future Research Direction	86
	References	87
A	List of Publications	95

List of Figures

- 3.1 Categorization of errors in critical cases, here each of the 60K, 40K, 24K, and 12K denotes the model trained on that particular lexicon. 23
- 3.2 Performance Comparison on Different Error Categories 26
- 3.3 Comparison in terms of WER and PER 27
- 3.4 Word Recognition Accuracy vs Iteration 28
- 3.5 Phoneme Recognition Accuracy vs Iteration 28
- 3.6 Negative Log Perplexity vs Iteration 29

- 4.1 Data Collection App 32
- 4.2 Data Collection App 35
- 4.3 Overview of Automated Corpus Preparation 36
- 4.4 Overview of Automatic Transcription 38
- 4.5 Histogram for percentage of longest common consecutive word sequence length between two transcriptions with respect to transcription from Google Speech API 41
- 4.6 Overview of our Text-to-Speech Architecture 43

- 5.1 Traditional ASR Overview 46
- 5.2 Traditional ASR Training 47
- 5.3 Traditional ASR Decoding 48
- 5.4 Overview of End-to-End Architecture 49
- 5.5 End-to-End Architecture 50
- 5.6 RNN Language Model 50

- 6.1 Contextual ASR Overview 53
- 6.2 System Overview 54
- 6.3 Contextual Rescoring in Client Device 56
- 6.4 Rescoring System 56
- 6.5 Contextual Corpus Extension 58
- 6.6 On Device Model Training 58
- 6.7 Effect of Context Weight w_3 61
- 6.8 Effect of CTC Weight w_1 61

6.9	Effect of Language Model Weight w_2	62
7.1	Baseline System	66
7.2	Semi-Supervised System	67
7.3	Overview of Encoding	68
7.4	t-SNE Visualization of Encoded Data	69
7.5	GED Loss	71
7.6	Supervised Training	74
7.7	Semi-supervised Retraining	75
7.8	Effect of CTC Weight w_1	76
7.9	Effect of Speech Text Ratio w_2	77
7.10	Effect of Supervised Loss Ratio w_3	78
7.11	Effect of Batch Size	78
8.1	Effect of Context Weight w_3	83
8.2	Effect of CTC Weight w_1	84
8.3	Effect of Language Model Weight w_2	84

List of Tables

2.1	Classification of Literature on G2P	8
2.2	Classification of Literature on corpus development	9
2.3	Classification of Literature on Contextual ASR	10
2.4	Different Approaches for Exploiting Unpaired Speech Data	11
2.5	Gap Analysis and Proposed Approach	12
3.1	Our Phoneme Symbols with Their Corresponding IPA Symbols	14
3.2	Error classification of 30K critical cases, here each of the four rightmost columns denotes the model trained on that particular lexicon.	21
3.3	Performance on Critical Cases	25
3.4	Performance Comparison In General	25
3.5	Performance comparison on different error categories. Here each of the three rightmost columns denotes the model trained on that particular lexicon.	26
3.6	Effectiveness of critical cases. Both lexicons are of size 60K. New Lexicon consists of 21K critical cases and 39K entries from Google lexicon.	29
4.1	Target Domain of Voice Command	30
4.2	Smartphone Operations Included	31
4.3	Apps Included	31
4.4	Google’s Crowd-Sourced Speech Corpus	31
4.5	Voice Command Corpus	35
4.6	Evaluation of Corpus by Iteration	41
4.7	Automatically Transcribed Corpus	42
4.8	Overall Speech Corpus	44
6.1	Performance comparison	61
6.2	WER for different Categories of Voice Command	62
7.1	Hyper-parameter Description	74
7.2	Performance Comparison with Baseline	75
7.3	Performance of Inter-Domain Loss	76
8.1	Speech Corpus	80

8.2	Evaluation of ASR performance	81
8.3	Decoding Speed	81
8.4	Speech Corpus	82
8.5	Performance on Voice Command Task	83
8.6	WER for different Categories of Voice Command	84

List of Algorithms

1	Algorithm for Compressing a Dictionary or Lexicon	20
2	Comparing a Generated Lexicon (gl) with Reference Lexicon (rl)	22
3	Procedure: <code>ocConfusion</code> (x, y) (Checks if open close vowel confusion)	23
4	Procedure: <code>removeVowel</code> (<code>phoneme_sequence</code>)	23
5	Iterative Algorithm for Transcription	40
6	Contextual Rescoring	57
7	Computation of the MMD loss	70
8	Computation of the GED loss	72

Abstract

Voice command recognition task commonly involves an Automatic Speech Recognition (ASR) system with context-specific optimization. Automatic Speech Recognition system development involves corpus resource development such as phoneme list, text corpus, word dictionary, phonetic dictionary, and speech corpus. These corpus resources are used to train speech recognition models. The performance of the speech recognition systems can be further improved by exploiting user and device-specific contexts. Context information for a specific smartphone user includes contact names, installed apps, songs, media files, location, recent search history, the content of the screen user is looking at, etc. The context information changes frequently so it is desired that the contextual model will be updated on-the-fly within the device. Traditional speech recognition systems usually consist of several individual components such as an acoustic model, a language model, a pronunciation dictionary, etc. So context-specific optimization can be achieved by tuning a particular component like the language model. Recently, end-to-end speech recognition architectures have been very effective in many speech recognition tasks. Incorporating context-specific optimization with the latest end-to-end speech recognition architectures requires a different approach. In this work, we focus on Bangla voice command recognition. We develop an ASR system for voice command recognition tasks and improve the performance further using context-specific optimization. In our work, we develop each linguistic resource in a way that considers language-specific characteristics of Bangla. We enrich our speech corpus with both domain-specific and domain-independent speech data. We also experiment with traditional and end-to-end speech recognition architectures. We propose a novel approach for context-specific optimization of voice commands. We also explore several other approaches for improving ASR performance such as synthetic speech corpus development and semi-supervised speech recognition.

Chapter 1

Introduction

In this section, we describe the objective of our research, motivation, problem definition, the outline of methodology, and our contribution.

1.1 Research Objective

In this work, we focus on voice command recognition in the context of standard Bangla language. The target domain for voice command recognition involves smartphone voice assistants, smart car voice assistants, home appliances, and office work accessories. We develop a large vocabulary Automatic Speech Recognition (ASR) System for Bangla language and perform domain-specific optimization by adding domain-specific speech corpus and by exploiting on-device user context during decoding.

1.2 Motivation

Bangla is spoken by 228 million native speakers and another 37 million second-language speakers. It is the fifth most spoken native language and the seventh most spoken language in the world. A lot of Bangla native speakers are using digital voice assistant technologies now. So a lot of research is required to improve Bangla voice assistant systems.

1.3 Present State

Currently, there are few publicly available Bangla speech corpus that is suitable for large vocabulary ASR. The largest speech corpus we found has around 220 hours of speech data. We did not find any corpus resource suitable for Bangla voice command recognition. All the previous literature on Bangla speech corpus development focus only on supervised speech corpus development. The ASR training strategy is also focused on supervised training.

1.4 Problem Definition

We focused on four research problems in our work. They are as follows:

Grapheme to Phoneme (G2P) Conversion Many ASR architectures use a G2P system to map the written representation of a word to its phonetic transcription. The input of this system is small manually verified lexicon and a list of words. The output is a lexicon that has phonetic transcription for all words in the word list.

Automated Speech Corpus Development The input for this system will be publicly available audio and text. The output is aligned speech corpus.

Semi-supervised ASR An ASR system is typically trained on an annotated speech corpus. It can also exploit an unpaired text corpus in the form of a language model. The semi-supervised ASR system aims to improve the performance of the ASR further by exploiting an unpaired audio corpus.

Context Specific Optimization The input of the system will be a set of candidate speech transcription and user context. The output is the speech transcription that is most relevant to the user context.

1.5 Outline of Methodology

The development of a voice command recognition system has several steps such as:

- Corpus Resource Development
- Speech Recognition Model Development
- Context-specific Optimization
- Other improvements

1.5.1 Corpus Resource Development

ASR systems are developed based on certain linguistic resources. They include phoneme list, text corpus, word dictionary, phonetic dictionary or lexicon, speech corpus, etc.

Phoneme List

A phoneme is a symbol that represents a sound. Traditional ASR systems usually convert acoustic signals to a sequence of phonemes. These phonemes are used to capture the pronunciation of words rather than their written representation. The phoneme list is chosen in a way that makes it possible to represent all pronunciations in a target language. The details of our phoneme list are described in section 3.1.

Text Corpus

A text corpus usually refers to a collection of processed text sentences. A text corpus is necessary for several important reasons. It is used to identify the most frequently used words in a language. A text corpus is also used to train a language model which can significantly improve the performance of an ASR system by providing the contextual relevance of a particular word in a transcription. Text corpus development typically involves web crawling, text cleaning, text normalization, etc. The details of our text corpus are described in section 3.2.

Word Dictionary

A word dictionary is a collection of words that covers the target domain for an ASR task. Typically, the most frequently used words in a language are kept in the word dictionary. The details of our word dictionary are described in section 3.2.

Phonetic Dictionary

The phonetic dictionary or lexicon maps the written representation of a word (grapheme sequence) to the phonetic representation (phoneme sequence). To prepare the lexicon, we need to have two things, a word dictionary whose phonetic transcription will be mapped and a set of phonemes that cover all pronunciations of the target language. The phonetic dictionary can contain 50,000 to 100,000 words, so it is very difficult to provide phonetic transcription of all these words manually. That is why manual transcription is done for a portion of the words. Then a Grapheme-to-phoneme (G2P) conversion system is trained on the manually transcribed lexicon. The rest of the words in the dictionary are transcribed using the G2P system. The details of our phonetic dictionary can be found in section 3.3.

Speech Corpus

Speech corpus refers to a collection of audio files with corresponding text transcriptions. The duration of each speech segment is kept under 35 seconds [1]. The speech corpus also contains some additional information such as the gender of speakers, recording environment, etc. Speech

corpus development is one of the most important tasks related to ASR system development. So it is discussed separately in Chapter 4.

1.5.2 Speech Recognition Model Development

There are several speech recognition architectures that learn to convert speech to text by exploiting the corpus resources. We experiment on both traditional and end-to-end architectures.

Traditional ASR System

Traditional ASR system has several components such as acoustic model, language model, etc. It has two basic steps for speech to text conversion. At first, the system absorbs the speech feature sequence and produces a phoneme sequence that best describes the pronunciation. In the next step, the phoneme sequences are converted to a sequence of words using the phonetic dictionary. During this step, the language model provides important information regarding the word context with respect to the other words in that transcription. The details of our traditional ASR system are described in section 5.1.

End-to-end ASR System

Unlike the traditional ASR system, the end-to-end system does not produce any intermediate phoneme like representations. Rather, it directly tries to convert speech to text using a single neural network architecture. The details of our end-to-end ASR system is described in section 5.2.

1.5.3 Context Specific Optimization

Voice command recognition task commonly involves an Automatic Speech Recognition (ASR) system with context-specific optimization. Context information for a specific smartphone user includes contact names, installed apps, songs, media files, location, recent search history, the content of the screen user is looking at, etc. The performance of the voice command recognition system can be significantly improved by exploiting the user context and device context. Our approach for context-specific optimization of Bangla voice command recognition task is described in Chapter 6.

1.5.4 Other Improvements

In our work, we try to address some additional research problems such as handling out-of-vocabulary word problem, the use of unpaired audio data, etc.

Handling Out-of-vocabulary Problem

An ASR system may need to transcribe speech containing certain words that never occurred in the speech corpus the system was trained on. These words are considered out-of-vocabulary (OOV) words. In some cases, the presence of OOV words severely hampers the ASR performance. In those cases, it needs to be addressed.

Semi-supervised ASR Training

Usually, ASR systems are trained on speech corpus where each speech segment is paired with corresponding text transcription. Large unpaired text corpus can also be exploited effectively by training a language model and incorporating language model scores with ASR model scores during decoding time. But the current ASR architectures are unable to exploit large unpaired audio corpus. In our work, we exploit large unpaired audio data using a semi-supervised training method. This approach is described in Chapter 7.

1.6 Our Contribution

Our contributions in this study are as follows:

- We have developed an improved lexicon that has around 100K most frequently used Bangla words and considers critical cases for G2P conversion in Bangla language. This lexicon can significantly improve the performance of ASR systems that depend on lexicon.
- We have developed a Bangla Speech Corpus containing both domain-independent and domain-specific utterances. The overall size of the corpus is 1010 hours. The domain-independent corpus contains 960 hours of data and the domain-specific corpus has 50 hours of data.
- We have developed a Bangla text corpus containing 10 million unique Bangla sentences. A language model trained on this text corpus significantly improves the ASR performance.
- We have developed a context-annotated voice command corpus. The corpus has 1700 voice command templates and includes all necessary voice commands related to smartphones, home appliances, automotive and office work accessories.
- We propose a novel approach for context-specific optimization of voice commands which is based on multi-label topic modeling.
- We propose a novel approach for semi-supervised ASR training that can improve ASR performance by exploiting unpaired audio data.

- We have solved the out-of-vocabulary problem for Bangla ASR task by using synthetic speech corpus generation.

Chapter 2

Literature Review

In this section, we provide a literature review for various ASR related components we worked on such as G2P conversion, automatic Speech Corpus Generation, context-specific optimization, semi-supervised training, etc.

2.1 G2P Conversion

The G2P system is used to find phonetic transcription for a word from its written representation. Table 2.1 shows the summary of previous approaches for G2P conversion.

The research works for G2P in English are quite extensive. [2] use joint maximum entropy n-gram model and conditional maximum entropy model for G2P conversion in English. [3] utilize Recurrent Neural Network (RNN) with bi-directional LSTM (Long Short Term Memory) for G2P and achieve 5.45% Phoneme Error Rate (PER) on CMU dictionary [4]. [5] show comparisons among various machine learning algorithms for G2P in Burmese language. [6] develop a joint-sequence model for G2P. Joint sequence n-gram models aim to discover joint vocabulary consisting of graphemes and phonemes through the alignment of graphemes and phonemes. [7,8] are other prominent works working on this model. Neural sequence to sequence models are popular for G2P conversion. Some prominent works on such models are: [3,9–19]. Most of the works related to G2P conversion in Bangla language follow rule-based approach. The rule-based approach of [20] provides an accuracy of 97.01% on a previously seen corpus containing 736 words, but the system’s accuracy is 81.95% on a previously unobserved corpus containing 8399 words. This work was extended by [21] describing 3880 rules with an accuracy of 89.48% on another corpus. [22] discuss a rule-based approach considering several information: parts-of-speech, subsequent context, etc. Their work describes only 21 rules and provides an accuracy of 91.48% on a corpus of 9294 words. [23] provide a heuristic for G2P that takes into account parts-of-speech, orthographic, and contextual information. Their work provides 70% accuracy on a corpus containing 755 words. A prominent work for data-driven

Literature	Language	Approach	Method
[3, 9–19]	English	Data-driven	Neural seq2seq
[2]	English	Data-driven	maximum entropy n-gram
[6–8]	English	Data-driven	Joint sequence n-gram
[20, 21]	Bangla	Rule-based	Phonetic Rules
[22]	Bangla	Rule-based	Phonetic Rules, parts-of-speech, subsequent context
[23]	Bangla	Rule-based	Phonetic Rules, parts-of-speech, orthographic, contextual information
[24]	Bangla	Data-driven	Neural seq2seq
[25]	Bangla	Data-driven	conditional random field

Table 2.1: Classification of Literature on G2P

G2P in Bangla language is by Google ([24]). They develop a lexicon and achieve word-level accuracy of 81.5%. [25] use conditional random field for G2P in Bangla. They report 14.88% phoneme error rate on Google lexicon. Another line of research deals with G2P conversion for more than one language. Such works include: [26], [27], [28], and [29].

The limitation of previous works on Bangla G2P is that most of them are rule-based. Rule-based G2P systems are not suitable for accurate G2P conversion for large dictionary size. [24] propose a data-driven G2P system that is trained on a manually transcribed lexicon of 37000 words. But the words in the manually transcribed lexicon are chosen according to frequency only. We will show that significant performance improvement can be made by choosing the words in the training set optimally.

2.2 Speech Corpus Preparation

In various languages, researchers have explored different methods for developing speech corpus. Jang and Hauptmann [30] develop a speech corpus from captioned multimedia speech. Lakomkin et al. [31] develop a tool to automatically construct data set for speech recognition from YouTube videos containing transcriptions. Panayotov et al. [1] present 1000 hours of speech corpus for English by aligning texts and audio files of audiobooks. Mansikkaniemi et al. [32] and Helgadóttir et al. [33] use similar alignment techniques to develop speech corpus from recordings and transcriptions from parliamentary speech. Patel et al. [34] build a data collection tool and collect around 100 hours of reading speech data in Manipuri Language.

Compared to other languages, research works on developing speech corpus for Bangla language are quite limited. Nahid et al. [35] discuss the development of Bangla real number audio corpus. The recordings were completed in a supervised environment and volunteers were given scripts to read. Khan and Sobhan [36] develop a speech corpus containing only isolated words for Bangla. All recordings were done in a laboratory. In another work of them, Khan and Sobhan [37]

Paper	Data Source	Approach	Approach Details	Hours
[1]	Audio Book			1000
[30]	Captioned Multimedia Speech			131.4
[31]	Captioned Youtube Speech	Automated	Forced Alignment	200
[32]	Parliamentary speech			1550
[33]	Parliamentary speech			542
[34]	Telephonic speech	Automated	Speech-to-text Keyword search Speaker diarization	100
[35]	Bangla real numbers			4
[36]	Bangla isolated words	Supervised	Record read speech	375
[37]	Bangla connected words			62
[38]	Crowd-sourced Bangla speech	Interactive app	Record read speech	220

Table 2.2: Classification of Literature on corpus development

develop a speech corpus of connected words for Bangla. Researchers from Google [38] prepare speech corpora for Bangla and four other languages using interactive mobile application [39]. They develop 229 hours of speech corpus for Bangla.

The limitation of the previous works on Bangla speech corpus development is that all the previous works adopt supervised speech corpus development approach. This is not always adequate for developing a speech corpus of the required size. Our work differs from the previous works as none of these works deals with the automatic preparation of speech corpus in Bangla language using existing audio and text data. We also prepare a separate domain-specific speech corpus for Bangla voice command recognition task.

2.3 Context-specific Optimization

In various languages, researchers have used contextual information to increase the performance of voice recognition systems. Table 2.3 shows the summary of previous approaches.

[40] present an online approach for adjusting language model (LM) weights of n-grams corresponding to a specific context. [41] describe a composition based on-the-fly re-scoring mechanism to employ contextual language models in a speech recognition system. [42] use Named-Entity Recognition within the automatic speech recognition word lattice for identifying contextually related paths. They report that their approach minimizes Word Error Rate (WER) by 12.0% on a media playing commands data set. [43] provide a mechanism to learn contextual information in an unsupervised manner and for building automatically contextually biased models. [44] discuss two interpolation methods to merge contextual information with knowledge from a general language model. [45] consider contextual information during beam search in an end-to-end speech recognition system. [46] discuss contextual recurrent neural network language model. They consider a contextual input vector for each word of a sentence.

Literature	ASR Architecture	Approach
[40, 41]	Traditional	contextual LM, N-gram
[42]	Traditional	Named-Entity Recognition in Word Lattice, N-gram
[43]	Traditional	Unsupervised context learning, N-gram biasing
[44]	Traditional	Merging contextual and general LMs, N-gram
[45]	End-to-end	N-gram
[46]	End-to-end	contextual RNNLM
[47]	End-to-end	class based LM, N-gram

Table 2.3: Classification of Literature on Contextual ASR

[47] use class-based language models that provide contextual information during decoding in an end-to-end speech recognition system. Moreover, [48] addresses an end-to-end ASR for low-resource multilingual ASR context. [49] develop an end-to-end automatic speech recognition system that is situation informed. They consider speaker gender, conversational history, etc. to develop the situation-informed system. [50] develop an end-to-end speech recognition system that considers dialog context.

We have not found any research work that focuses on considering contextual information for voice command recognition in Bangla language. For other languages, all of the approaches used are variations of n-gram based model for context detection. But the n-gram based approach is too rigid. It is not robust to synonymous, missing, or misplaced words. All possible synonyms and n-gram variations need to be present in the contextual corpus. It also considers a lot of irrelevant word combinations as the contextual corpus gets bigger. We propose a multi-label topic modeling approach for context detection which has several advantages over the n-gram based approach. The topic modeling approach works on keywords which is more flexible and robust than the n-gram approach. A variable number of contexts can be easily handled with multi-label topic modeling.

2.4 Semi-supervised ASR Training

Researchers have explored different methods of exploiting unpaired speech data for speech recognition. Table 2.4 shows the summary of previous approaches.

[51] investigate large-scale semi-supervised training to improve acoustic models for automatic speech recognition. They provide an empirical analysis of semi-supervised training with respect to transcription quality, data quality, filtering, etc. [52] pre-train the encoder-decoder network with unpaired speech and text. They use a large amount of unpaired audio to pre-train the encoder and synthesized audio from the unpaired text to pre-train the decoder. [53], [54] integrate active

Literature	Architecture	Approach
[51]	Traditional	Semi-supervised training, acoustic model
[53, 54]	Traditional	active learning, semi-supervised training
[55]	Traditional	Multilingual data exploitation, acoustic model
[57]	Traditional	DNN, Graph based learning of acoustic model
[58]	Traditional	Sparse Auto-encoder
[59–61]	Traditional	Semi-supervised learning of acoustic model, lexicon
[63, 64]	End-to-end	Semi-supervised training with shared encoder

Table 2.4: Different Approaches for Exploiting Unpaired Speech Data

learning jointly with semi-supervised training in speech recognition system. [55] use transcribed multilingual data and semi-supervised training to circumvent the lack of sufficient training data for acoustic modeling. They train deep neural networks as data-driven feature front ends.

[56] use utterance-level and frame-level confidences for data selection during self-training. They find it beneficial to reduce the disproportion in amounts of paired and unpaired data by including the paired data several times in semi-supervised training. [57] describe the combination of deep neural networks and graph-based semi-supervised learning for acoustic modeling in speech recognition. [58] use a sparse auto-encoder to take advantage of both unlabelled and labeled data simultaneously through mini-batch stochastic gradient descent.

[59] try to improve the performance of a code-switching speech recognition system for Mandarin-English using semi-supervised training. They apply semi-supervised learning for lexicon learning as well as acoustic modeling. Similarly, [60] and [61] use untranscribed data for Luxembourgish & Lithuanian ASR respectively. [62] use a two-step training method to generalize the air traffic control speech recognizer. First, a baseline speech recognition system is trained using a paired speech corpus and it is used to transcribe publicly available unlabeled data. The transcribed data is then filtered based on confidence scores and is used to retrain the acoustic model.

Recently, semi-supervised training has been proposed in the context of end-to-end ASR. [63] propose a shared encoder architecture for speech and text inputs that can encode both data from their respective domain to a common intermediate domain. They combine speech-to-text and text-to-text mapping by using the shared network to improve speech-to-text mapping. They propose an inter-domain loss function based on Gaussian KL-divergence which represents the dissimilarity between the encoded features of speech and text data. They later proposed an inter-domain loss function based on Maximum Mean Discrepancy [64]. In both cases, they assume that the encoded speech features in the current minibatch are sampled from one distribution and encoded text features in the current minibatch are sampled from a second distribution. The inter-domain loss is calculated based on the discrepancy of these two distributions. This approach has some weaknesses. The performance of this system varies based on the chosen minibatch size. Moreover, this approach does not take into account the variance of the current encoded features

Research Problem	Gap	Proposed Solution
Grapheme to phoneme conversion	Conversion Mostly rule based for Bangla, verified lexicon for data driven approach has words with trivial G2P cases	Identify words with critical G2P rules, Verify most words automatically, Verify critical words manually
Automated speech corpus preparation	No work for automated corpus generation in Bangla, difficult to use forced alignment in Bangla	Combine speaker diarization, gender detection, silence detection and existing ASR system to extend the speech corpus, Speech synthesis for OOV
Semi-supervised ASR	No previous work in Bangla, Unsupervised loss not robust enough in previous end-to-end system	Unsupervised loss calculation in global context
Context Specific Optimization	No previous work in Bangla, N-gram too rigid, Too many irrelevant word combinations	Keyword based context identification and contextual relevance calculation

Table 2.5: Gap Analysis and Proposed Approach

in the global context. We solve both problems by introducing a new inter-domain loss function based on global encoding distance.

2.5 Summary of Gap Analysis

Table 2.5 shows the summary of gap analysis in the previous literature and our proposed approach for solving the research problems. We will discuss each of the proposed approach in detail in the following chapters.

Chapter 3

Linguistic Resource Preparation

In this chapter, we describe the preparation of different linguistic resources for the ASR system. Traditional ASR systems use a phonetic dictionary or lexicon to map a word to its phonetic transcription. To prepare the lexicon, we need to have two things, a word dictionary whose phonetic transcription will be mapped and a set of phonemes that cover all pronunciations of the target language. Typically, the most frequently used words in a language are kept in the word dictionary. The word dictionary can contain 50,000 to 100,000 words, so it is very difficult to provide phonetic transcription of all these words manually. That is why manual transcription is done for a portion of the words. Then a Grapheme-to-phoneme (G2P) conversion system is trained on the manually transcribed lexicon. The rest of the words in the dictionary are transcribed using the G2P system. The G2P system is also very helpful for finding grapheme-phoneme mapping of out-of-vocabulary words.

3.1 Phoneme List

Our Phoneme symbols are provided in Table 3.1. This table is a good reference for the 47 phoneme symbols that we have followed in this work and their corresponding International Phonetic Alphabet (IPA) symbols. Throughout the book, we use these 47 phoneme symbols, not the IPA symbols. There is a disagreement between linguists whether nasal vowels should be considered as separate phonemes [65]. We added nasal vowels in our phoneme list to differentiate between a word with its nasalized counterpart, such as the word কাঁদা(to cry) and কাদা(mud). Here, /a/, /e/, /u/, /i/, /o/, /O/, /E/, /an/, /en/, /un/, /in/, /on/, /On/, /En/ are normal vowels, /ew/, /ow/, /uw/, /iw/ are weak vowels, and the rest are consonants.

Phoneme	IPA	Phoneme	IPA	Phoneme	IPA	Phoneme	IPA
i	i	On	ɔ̃	D	ḍ	m	m
u	u	an	ã	Dh	ḍ ^h	r	r
e	e	k	k	t	t	R	ɽ
o	o	kh	k ^h	th	t ^h	l	l
E	ɛ	g	g	d	d	h	ɦ
O	ɔ	gh	g ^h	dh	d ^h	s	s
a	a	c	tʃ	p	p	sh	ʃ
in	ĩ	ch	tʃ ^h	ph	p ^h	iw	ĩ
un	ũ	j	dʒ	b	b	ew	ẽ
en	ẽ	jh	dʒ ^h	bh	b ^h	ow	õ
on	õ	T	t	N	ŋ	uw	u̇
En	æ̃	Th	t ^h	n	n		

Table 3.1: Our Phoneme Symbols with Their Corresponding IPA Symbols

3.2 Text Corpus and Word Dictionary Preparation

A text corpus is necessary for several important reasons. It is used to identify the most frequently used words in a language. A text corpus is also used to train a language model which can significantly improve the performance of an ASR system by providing the contextual relevance of a particular word in a transcription. Text corpus development typically involves web crawling, text cleaning, text normalization, etc.

3.2.1 Web Crawling

To get a hold of the contemporary usage of Bangla language, we do extensive crawling. We have prepared a crawling tool for open-domain sentence collection. We used the scrapy framework for the crawling tool. For every website, we created a unique spider that will crawl from that website. We used a web app for interfacing with the crawler. The web app is built in Django.

We crawled 42 websites of various Bangla newspapers, blogs, e-book libraries, Wikipedia, etc. covering various domains such as politics, economics, sports, drama, novel, stories, education, entertainment, general knowledge, history, etc. We tried to cover every domain and make sure there was very little repetition and the corpus covers most of the Bengali words we use every day. After collecting the text corpus using Web-Crawler, we parsed them into sentences. There were around 11 million sentences after crawling.

3.2.2 Text Cleaning

After collecting text from the open domain, we clean the texts using our text cleaning tool. We consider the following cases while cleaning the texts:

- Remove non-Bangla Sentences.
- Remove punctuation's and signs
- Replace it with blank space.
- Remove inconsistent lines.
- Remove duplicate lines.
- Sort sentences lexicographically.

3.2.3 Text Normalization

After cleaning the texts, we normalized those using the following rules:

- Numbers are converted to text
- If comma (,) is inside a number, will be replaced by null string, if anywhere else, will be replaced by single space
- Handling abbreviation, such as: “মোঃ”: “মোহাম্মদ”
- Some special numeric such as: “১ম”: “প্রথম”, “২য়”, “দ্বিতীয়”,
- Handling Percentage (%) sign, will be changed to: “শতাংশ”
- Replaced tab(s), multiple spaces with single space
- Handled decimal symbol (.), will be converted to: “দশমিক”, also the digits after decimal symbol will be converted digitwise: so, “১২৩.০৫৬” will be converted to: “এক শত তেইশ দশমিক শূন্য পাঁচ ছয়”
- If there is “সাল, সন” etc. after number, then it will be changed differently to capture how we naturally convert year: for example: “১৯৭১ সালের ২৫শে মার্চ” will be: “উনিশ শত একাত্তর সালের পঁচিশ শে মার্চ”, not: “এক হাজার নয় শত একাত্তর সালের”
- If a number starts with ‘0’ (zero), digit wise change (i.e., considering it as mobile number or number plate) so: “০১৭২৩৪৫৬” will be “শূন্য এক সাত দুই তিন চার পাঁচ ছয়”

- Hyphen (-) will mostly be ignored, but in certain situations, it will be converted to “থেকে”, such as: “২-৩ দিন” => “দুই থেকে তিন দিন” “২১ সেপ্টেম্বর - ২২ অক্টোবর” => “২১ সেপ্টেম্বর থেকে ২২ অক্টোবর”

After data cleaning and data normalization, we had about 10 million sentences in our text corpus.

3.2.4 Domain Specific Text Corpus

For close domain sentence collection, we tried to cover commands supported by existing voice assistants below:

- Google Assistant
- Google Home
- Amazon Alexa
- Siri
- Bixby
- Cortana

We explored the commands supported by these voice assistants in English and tried to add their Bangla equivalent commands in our corpus. Overall, we had 1700 unique voice command text in our corpus. We add these sentences to our text corpus. The collected sentences were from the following domains:

- Smartphone and Popular app commands
- Home appliance
- Office
- Automotive

3.2.5 Word Dictionary

After doing word frequency analysis of these sentences, we got around 1.7M unique Bangla words (tokens). We counted how many times each of the unique words appeared in those sentences. We then consider the most frequent 100K words and aim to identify the critical cases for phonetic transcription among these most frequent words. And these words constitute our word dictionary for Bangla language. For the voice command-specific sentences, we add each word to our word dictionary regardless of their frequency.

3.3 G2P Training and Lexicon Preparation

Grapheme to phoneme (G2P) conversion provides a mapping between a word and its pronunciation. Such mapping provides the opportunity for a non-native person to learn the correct pronunciation of words of a foreign language. Moreover, in modern Text to Speech (TTS) and Automatic Speech Recognition (ASR) systems, G2P conversion is an integral task. The task of G2P conversion is generally language-specific due to language-specific conventions, rules, pronunciation constraints, etc. In our work, we focus on Modern Standard Bangla. An example of G2P conversion in Bangla language: phonetic transcription of অনুশীলন (practice) is /o n u s h i l O n/.

The simplest means of G2P conversion is to build up a lexicon or dictionary containing the mapping from words to their corresponding pronunciations. However, it fails to provide pronunciations for unknown words and the inclusion of newer words increases memory requirement. In another approach [20], there are predefined rules for the conversion of a word to its pronunciation. Though such a rule-based approach can work for any word, the system becomes complex when it tries to formulate rules for incorporating all irregularities of pronunciation in a language.

These approaches are not feasible for large-scale G2P conversion which is necessary for any modern TTS or ASR system. Data-driven machine learning approaches have great potential in such large-scale G2P conversion [12]. In such an approach, a machine learning model predicts the phoneme conversion of a grapheme, being trained on a lexicon. A predominant work following such approach in Bangla language is by Google [24], where they train their system using 37K words and achieve word-level accuracy of 81.5%. However, a system trained on their lexicon will face several shortcomings, such as কাঁদা(mud) and কাঁদা(to cry) are pronounced differently but will have the same phoneme representation in their system as: /k a d a/. Similarly, পরী(fairy) and পড়ি(to read) are pronounced differently but will have the same phoneme representation in their system as: /p o r i/. Moreover, G2P system trained on their lexicon performs poorly on our identified critical cases from the most frequent 100K words (Table 3.3).

Being motivated to increase the accuracy of grapheme to phoneme conversion in Bangla language, which will also perform well for critical inputs, we have developed a customized and robust G2P system for Bangla language.

Our major contributions are as follows:

- (i) We identify and categorize the critical cases for grapheme to phoneme (G2P) conversion in Bangla language by analyzing the most frequent 100K words.
- (ii) We enrich the training lexicon for developing a robust G2P conversion system in Bangla language that performs much better for critical cases compared to other state-of-the-art

G2P systems.

- (iii) We perform phonetic transcriptions considering nasal vowels as separate phonemes.
- (iv) We perform extensive simulations on a large-scale dataset and show that our methodology outperforms other state-of-the-art approaches for G2P conversion in Bangla language by providing word-level accuracy of 90.2%.

3.3.1 Previous Works on G2P

The research works for G2P in English are quite extensive. [2] investigate machine learning based systems for G2P in English. They experiment with joint maximum entropy n-gram model, conditional maximum entropy model, etc. [3] utilize bi-directional LSTM (Long Short Term Memory) recurrent neural network for G2P and achieve 5.45% PER on CMU dictionary [4]. [5] show comparisons among various machine learning algorithms for G2P in Burmese language. Joint sequence n-gram models aim to discover joint vocabulary consisting of graphemes and phonemes through the alignment of graphemes and phonemes. [6] develop a joint-sequence model for G2P. [7], [8] are other prominent works working on this model. Neural sequence to sequence models are popular for G2P conversion. Some prominent works on such models are: [9], [10], [11], [3], [10], [12], [3], [13], [14], [15], [16], [17], [18], and [19]. Again, another line of research deals with G2P conversion for more than one language. Such works include: [26], [27], [28], and [29].

Most of the works related to G2P conversion that are focused on Bangla language, follow rule-based approach. The rule-based approach of [20] provides an accuracy of 97.01% on a previously seen corpus containing 736 words, but the system's accuracy is 81.95% on a previously unobserved corpus containing 8399 words. This work was extended by [21] describing 3880 rules with an accuracy of 89.48% on another corpus. [22] discuss a rule-based approach considering several information: parts-of-speech, subsequent context, etc. Their work describes only 21 rules and provides an accuracy of 91.48% on a corpus of 9294 words. [23] provide a heuristic for G2P that takes into account parts-of-speech, orthographic, and contextual information. Their work provides 70% accuracy on a corpus containing 755 words. A prominent work for data-driven G2P in Bangla language is by Google ([24]). They develop a lexicon and achieve word-level accuracy of 81.5%. [25] use conditional random field for G2P in Bangla. They report 14.88% phoneme error rate on Google lexicon.

3.3.2 Non-Trivial Cases for Transcription

We envision developing a robust G2P system that will perform reasonably well on any word in Bangla language. A G2P system that performs well on the most frequent words, should

also do well on other words. With this motivation, we focus on increasing accuracy on the most frequent words. Especially, we are concerned about those words that are among the most frequent words but non-trivial or critical for phonetic transcription, i.e., current state-of-the-art methodologies perform poorly on these critical words. We investigate identifying and categorizing such non-trivial or critical cases so that future research works can give special focus on developing methods for improving phonetic transcriptions of these critical words.

Identifying the Critical Cases for Transcription

After changing the Google lexicon (of size 60K (around)) according to our phoneme symbols (Table 3.1), we prepare 4 versions of Google’s lexicon of size 12K, 24K, 40K, and 60K respectively for identifying the critical cases for phonetic transcription. Algorithm 1 shows prefix comparing algorithm that we use for compressing a phonetic lexicon or dictionary of grapheme sequence to phoneme sequence. The algorithm matches the prefix of consecutive words (grapheme sequence) of a sorted dictionary (sorted according to ascending order of grapheme sequence of a word) and keeps a word (with its corresponding phoneme sequence) only if it does not share its prefix with any other words. We run the algorithm successively 3 times, i.e., we use the destination dictionary of one iteration as the source dictionary of the next iteration. Each iteration produces a minimized version of the basic lexicon (Google lexicon). After 3 iterations, the dictionary does not get any more compressed. We find the phonetic transcriptions of each of the 100K most frequent words using models trained on each of the 4 versions of Google’s lexicon (basic + 3 minimized). So, from 4 models (each model trained on a version of the basic Google lexicon), we get 4 sets of transcriptions for the most frequent 100K words. For most of the words (around 70K words), we observe that the phonetic transcriptions are exactly the same in each of the 4 set. However, for the remaining 30K words (29105 words to be exact), we observe that at least one set provides different transcription. We take these 30K words to be the critical cases. Our intuition is that if two G2P systems: one trained on a smaller version of the basic lexicon, and another trained on a larger version of the basic lexicon provide the same transcription for a word, then the word is a trivial case for phonetic transcription. We then manually verify the phonetic transcriptions of these 30K words taking help from 3 linguists and following [66], and consider these 30K words as critical cases for phonetic transcription.

Categorizing the Critical Cases

We categorize the critical cases into 7 categories and observe the distribution of the critical transcriptions into these 7 categories. These 7 categories capture most of the errors. The categories are:

- **Open Close Vowel Confusion:** G2P system provides pronunciation as close vowel that should be pronounced as open vowel ideally, and vice-versa. For example, correct

Algorithm 1 Algorithm for Compressing a Dictionary or Lexicon

```

1:  $sd \leftarrow$  sorted sourceDictionary
2:  $dd \leftarrow$  sorted destinationDictionary
3:  $a.grs \leftarrow$  grapheme sequence of lexicon
4:     entry  $a$ 
5: add  $sd[0]$  to  $dd$ 
6:  $i = 1$ 
7: while  $i \neq \text{length}(sd)$  do
8:      $pw = sd[i - 1]$ 
9:      $cw = sd[i]$ 
10:    if  $\text{length}(pw.grs) \geq 3$  &  $pw.grs$  is prefix of  $cw.grs$  then
11:        continue
12:    else
13:        add  $cw$  to  $dd$ 
14:     $i \leftarrow i + 1$ 

```

phoneme of ব্যাঙ (frog) is /b E n g/, but if the G2P system provides output /b e n g/, then it is an error under this category as in the place of an open vowel (here, /E/), the G2P system is giving close vowel (here, /e/).

- **Inherent Vowel Confusion:** G2P system does not provide inherent vowel as output where there should be an inherent vowel ideally. For example, correct phoneme of সকাল (morning) is /sh O k a l/, but if the G2P system provides output /sh k a l/, then it is an error under this category as the output of G2P does not give the inherent vowel (here, /O/).
- **Diphthong Confusion:** G2P system does not provide falling diphthong in output where there should be a falling diphthong ideally. Or, the system does not provide a rising diphthong in output where there should be a rising diphthong ideally. For example, correct phoneme of সেই (friend) is /sh o iw/, but if the G2P system provides output /sh o i/, then it is an error under this category as the output of G2P does not capture the falling diphthong (here, /o iw/).
- **s or sh Confusion:** G2P system provides /s/ in phonetic transcription, where there should be /sh/, and vice-versa. For example, correct phoneme sequence of সংগঠন (organization) is /sh O N g O Th o n/, but if the G2P system provides output /s O N g O Th o n/, then it is an error under this category as the output of G2P gives /s/ in place of /sh/.
- **s or ch Confusion:** G2P system provides /s/ in phonetic transcription, where there should be /ch/, and vice-versa. For example, correct phoneme sequence of ছাতা (umbrella) is /ch a t a/, but if G2P system provides output /s a t a/, then it is an error under this category as the output of G2P gives /s/ in place of /ch/.
- **Nasal Confusion:** G2P system does not provide any nasal vowel where there should be a nasal vowel, and vice-versa. For example, correct phoneme sequence of চাঁদ (moon) is /c

Error Type	60K	40K	24K	12K
total error	6415	7222	8087	10400
Open Close Confusion (%)	32.0	30.7	34.8	23.6
Inherent Vowel Confusion (%)	28.4	36.7	32.0	40.1
s or sh confusion (%)	14.6	15.3	14.2	12.8
Diphthong confusion (%)	11.6	9.8	7.6	9.4
Other Vowel Confusion (%)	2.1	1.3	4.2	3.1
s or ch confusion (%)	0.8	0.7	0.2	0.5
Nasal Confusion (%)	0.2	0.1	0	0
Other Error (%)	10.4	5.4	7.1	10.6

Table 3.2: Error classification of 30K critical cases, here each of the four rightmost columns denotes the model trained on that particular lexicon.

an d/, but if the G2P system provides output /c a d/, then it is an error under this category as the output of G2P gives /a/ in place of /an/.

- **Other Vowel Confusion:** The G2P system provides a completely different vowel than the corresponding vowel that should ideally be in that position of the phoneme sequence. Note that, in the other error categories, for each position in the phoneme sequence, the generated and ideal phonemes were somehow related. But in this category, at a specific position of the phoneme sequence, the generated and ideal phonemes are completely different. For example, correct phoneme sequence of অধ্যবসায় (perseverance) is /o d dh o b O sh a ew/, but if the G2P system provides output /o d dh a b O sh a ew/, then it is an error under this category as the output of G2P gives /a/ in place of /o/ (fourth phoneme).

Algorithm 2 compares a machine-generated lexicon with a reference lexicon (manually verified), where both the lexicons have the same grapheme sequences, but the corresponding phoneme sequences may be different. This algorithm counts how many errors of each category are there in the machine-generated lexicon. The algorithm takes each entry of the generated lexicon and increases the count of the corresponding error category (if an error is present there).

We train the attention mechanism based Transformer model on each of the 4 lexicons and get 4 G2P models. We find the phoneme representation of 30K critical cases using each of the 4 G2P models. Using Algorithm 2, we count the errors of each category for each of the 4 models. We report the results in Table 3.2 and Figure 3.1. Here, the other error denotes the errors that are not captured by these 7 categories. We see from these results that most of the errors are under Open Close vowel, s or sh, Diphthong, and Inherent Vowel confusions.

3.3.3 Developing an Improved G2P System for Bangla Language

We develop an improved lexicon and use two machine learning-based models trained on our lexicon to develop an improved G2P system for Bangla language. For developing an improved

Algorithm 2 Comparing a Generated Lexicon (gl) with Reference Lexicon (rl)

```

1:  $N \leftarrow$  total number of entries in each lexicon
2:  $A, B, C, D, E, F, G$  are Open Close Vowel, s or sh, s or ch, Nasal, Diphthong, Other
   Vowel, and Inherent confusions, respectively, all initially zero
3:  $H$  denotes other errors not captured by the 7 categories, initially zero
4:  $vl$  and  $wl$  are lists of vowels and weak vowels respectively
5:  $a.phs \leftarrow$  phoneme sequence of lexicon
6:     entry  $a$ 
7:  $i \leftarrow 0$ 
8: while  $i \neq N$  do
9:      $g = gl[i].phs$ 
10:     $r = rl[i].phs$ 
11:     $M = \min(\text{length}(g), \text{length}(r))$ 
12:     $j \leftarrow 0$ 
13:    while  $j \neq M$  do
14:         $x = g[j]$ 
15:         $y = r[j]$ 
16:        if  $x = y$  then
17:            continue
18:         $(x, y) \leftarrow \text{sorted}(x, y)$ 
19:         $Total\_error = Total\_error + 1$ 
20:        if  $ocConfusion(x, y)$  then
21:             $A \leftarrow A + 1$ 
22:        else if  $(x, y) = ("s", "sh")$  then
23:             $B \leftarrow B + 1$ 
24:        else if  $(x, y) = ("ch", "s")$  then
25:             $C \leftarrow C + 1$ 
26:        else if  $x + "n" = y$  then
27:             $D \leftarrow D + 1$ 
28:        else if  $x$  in  $wl$  or  $y$  in  $wl$  then
29:             $E \leftarrow E + 1$ 
30:        else if  $x$  in  $vl$  and  $y$  in  $vl$  then
31:             $F \leftarrow F + 1$ 
32:        else if  $removeVowel(g) = removeVowel(r)$  then
33:             $G \leftarrow G + 1$ 
34:        else
35:             $H \leftarrow H + 1$ 
36:         $j \leftarrow j + 1$ 
37:     $i \leftarrow i + 1$ 

```

Algorithm 3 Procedure: `ocConfusion` (x, y) (Checks if open close vowel confusion)

```

1:  $ocSet \leftarrow [(\text{"O"}, \text{"o"}), (\text{"E"}, \text{"e"}),$ 
2:    $(\text{"On"}, \text{"on"}), (\text{"En"}, \text{"en"})]$ 
3: if ( $x, y$ ) in  $ocSet$  then
4:   return True
5: else
6:   return False

```

Algorithm 4 Procedure: `removeVowel` (`phoneme_sequence`)

```

1: return phoneme_sequence removing all vowels from it

```

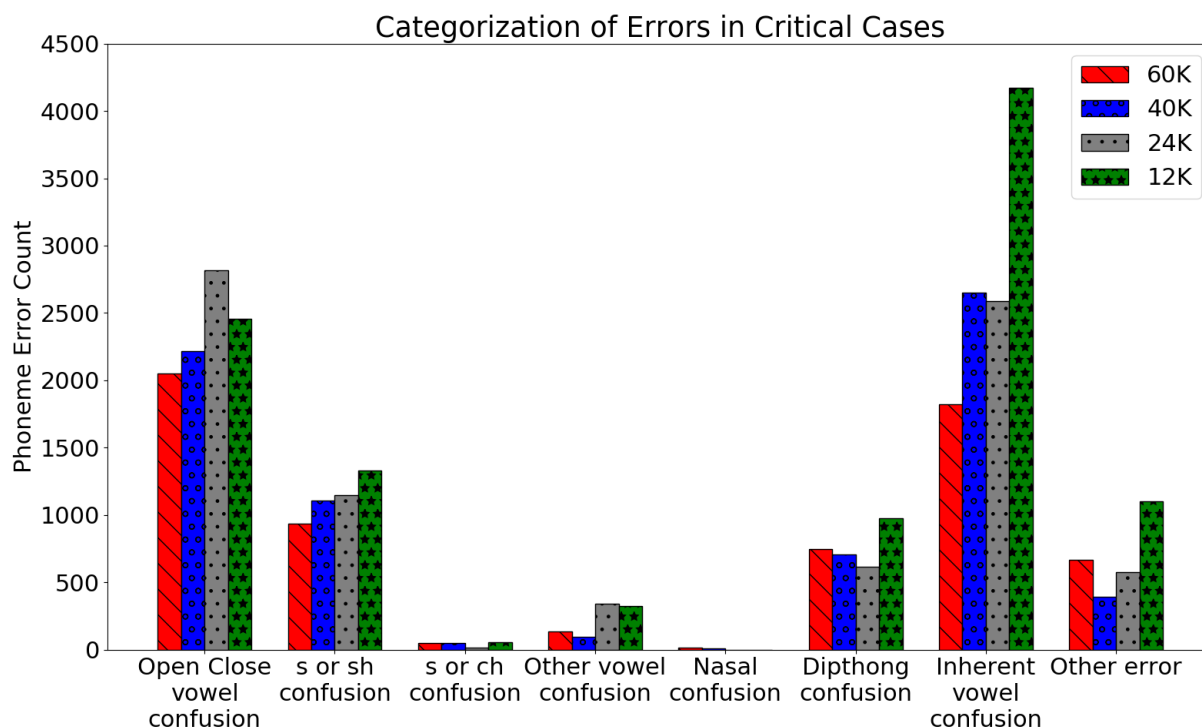


Figure 3.1: Categorization of errors in critical cases, here each of the 60K, 40K, 24K, and 12K denotes the model trained on that particular lexicon.

lexicon, we include with Google’s lexicon the manually verified 30K non-trivial or critical entries. Also, we include the 70K entries in which all of the 4 models (trained on each of the 4 versions of Google lexicon) unanimously agreed. Our lexicon consists of around 100K entries.

3.3.4 G2P Models

We use Neural Sequence to Sequence models. In these models, a conditional distribution of a sequence (here, phoneme sequence) is learned conditioned on another sequence (here, grapheme sequence). We train two following sequence-to-sequence models on our lexicon for G2P conversion:

LSTM-RNN: This is a plain Sequence to Sequence model that incorporates an encoder and decoder mechanism. A recurrent Neural Network (RNN) is usually utilized in encoder and decoder design. For addressing the vanishing gradient problem in RNN, Long-Short Term Memory (LSTM) [67] is used. We follow [3] for implementation.

Transformer Model: Transformer Model uses attention mechanism. Attention mechanism provides an improvement upon plain Sequence to Sequence by easing the flow of information from source sequence to destination sequence. We follow [19] for implementation.

We show the performance of both of these models in Section 3.3.5. We observe that Transformer Model provides higher token-level accuracy (lower Word Error Rate) than LSTM-RNN.

3.3.5 Experimental Results

We run extensive simulations and use two measures for evaluating the performances of the G2P systems:

Word Error Rate (WER): For calculating Word Error Rate (WER), we use the following formula:

$$\text{WER} = \frac{E}{T}$$

where E denotes the number of words that disagree on their generated phoneme sequence and reference phoneme sequence, and T denotes the total number of words.

Phoneme Error Rate (PER): For calculating Phoneme Error Rate (PER), we use the following formula:

$$\text{PER} = \frac{I + S + D}{T}$$

where I , S , D denote respectively the total number of insertion, substitution, and deletion operations needed for all the words to align the generated phoneme sequence with the reference

Lexicon	Model	WER(%)	PER(%)
Google	LSTM-RNN	25.7	3.26
	Transformer Model	23.6	2.71

Table 3.3: Performance on Critical Cases

Lexicon	Model	WER (%)	PER (%)
Google	LSTM-RNN	17.1	2.32
	Transformer Model	14.8	1.88
Our Lexicon	LSTM-RNN	10.5	1.42
	Transformer Model	9.8	1.33

Table 3.4: Performance Comparison In General

phoneme sequence for each word. T denotes the total number of phonemes present in all the words.

Our best performing model is Transformer Model. We use a batch size of 4096. Our neural network has 3 hidden layers, each containing 256 nodes. We use a computer having 8GB RAM, Intel Core i7 CPU, and Nvidia Geforce 1050 GPU for running all of the simulations. For each model, we run the simulations for around 110K iterations taking around 5 hours.

Performance on Critical Cases

We report the experiment results of Google’s lexicon on critical cases in Table 3.3. We do not report our lexicon here as critical cases are already included in our lexicon.

Performance Comparison In General

For comparing the performances of models trained on our lexicon and Google’s lexicon, we randomly take 9000 entries from our manually verified 30K critical cases as the test set. We use this test set for evaluating all the models. Though our actual lexicon contains these 9000 entries, we do not keep them in our lexicon while doing the experiments to fairly evaluate the performances of the lexicons. For both lexicons, we keep 90% of the lexicon in the train set and the remaining 10% in the validation set. Table 3.4 shows the result. Models trained on our lexicon outperforms those trained on Google’s lexicon by a significant margin. Moreover, Transformer Model performs better than LSTM-RNN.

Figure 3.2 and Table 3.5 categorize the errors of systems trained on 3 types of lexicons (Romanized version of our lexicon is discussed in section 3.3.5) by using Algorithm 2. Here, we report the results of the Transformer Model only as it has been better performing than LSTM-RNN in our experiments. We observe most of the errors are related to Open Close vowel, s or sh, Diphthong, and Inherent Vowel confusions - this finding also conforms to Figure 3.1 and Table 3.2, which were error categorization of critical cases.

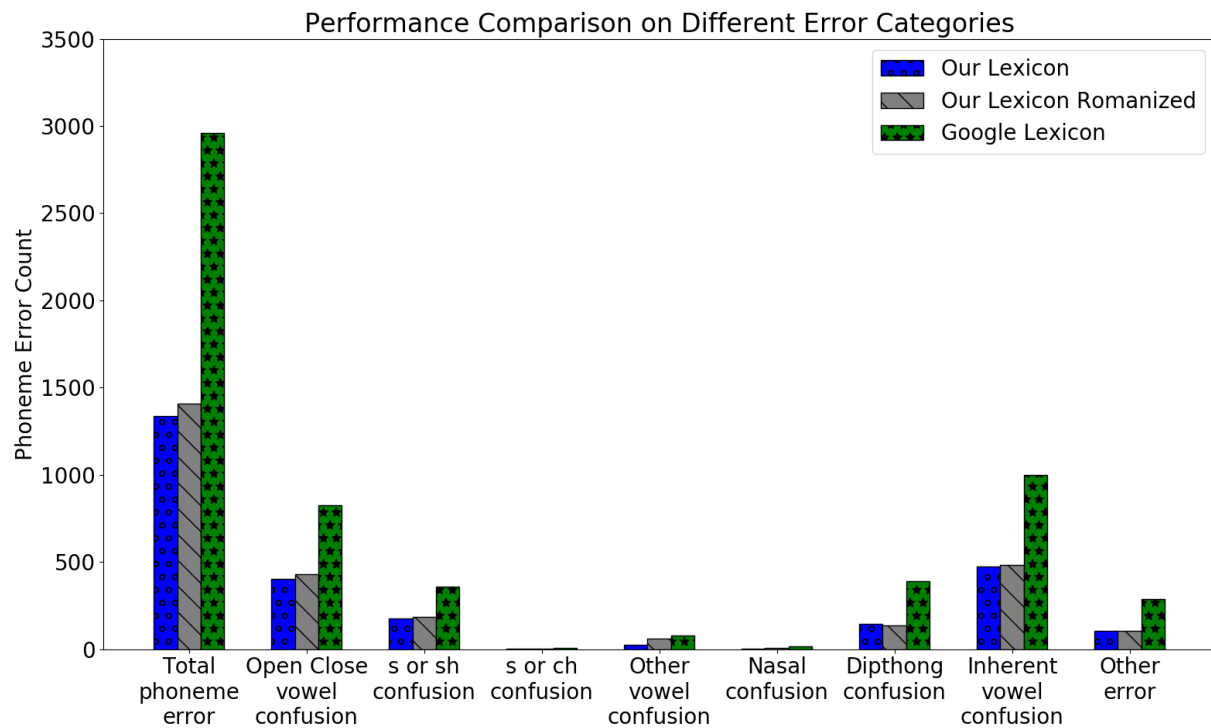


Figure 3.2: Performance Comparison on Different Error Categories

Error Type	Our Lexicon	Romanized Lexicon	Google Lexicon
Total Error	1337	1406	2961
Inherent Vowel Confusion (%)	35.6	34.5	33.8
Open Close Confusion (%)	30.1	30.4	27.9
s or sh confusion (%)	13.3	13.0	12.1
Diphthong confusion (%)	10.7	9.8	13.1
Other Vowel Confusion (%)	1.9	4.3	2.7
s or ch confusion (%)	0.3	0.2	0.2
Nasal Confusion (%)	0.2	0.43	0.6
Other Error (%)	7.9	7.4	9.7

Table 3.5: Performance comparison on different error categories. Here each of the three rightmost columns denotes the model trained on that particular lexicon.

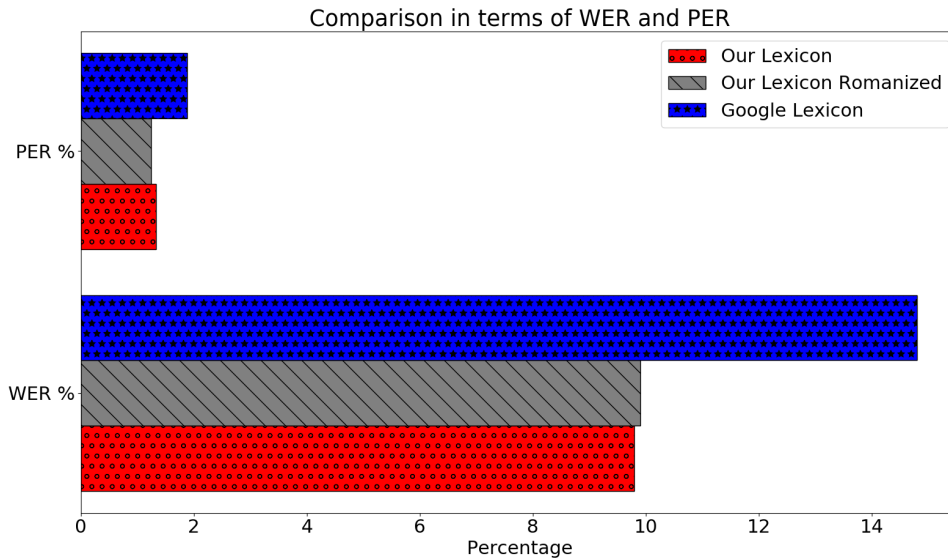


Figure 3.3: Comparison in terms of WER and PER

Effect of Romanization

For doing experiments on the effect of romanization on G2P conversion, we romanized all grapheme sequences in our lexicon to prepare a romanized counterpart of our lexicon. During romanization, each grapheme symbol in Bangla is replaced with a single English letter except that if a consonant grapheme is not followed by a vowel grapheme, “O” was added after the romanized symbol of that consonant as the roman symbol for “অ”, which is usually inherently pronounced in such cases. All the symbols used for romanization were completely disjoint to avoid any ambiguity in the lexicon. Figure 3.3 shows the WER and PER of systems trained on 3 types of lexicons. Here, we report the results of the Transformer Model only as it has been better performing than LSTM-RNN in our experiments. Both versions of our lexicon perform better (lower WER and lower PER) than Google’s lexicon. Also, romanization does not significantly increase or decrease the performance.

Figures 3.4 and 3.5 show respectively the Word Recognition Accuracy ($1 - \text{WER}$) and Phoneme Recognition Accuracy ($1 - \text{PER}$) with respect to number of iterations run during simulation. Both versions of our lexicon perform better (higher Word Recognition Accuracy and higher Phoneme Recognition Accuracy) than Google’s lexicon. Figure 3.6 shows Negative Log Perplexity vs number of iterations. Both versions of our lexicon provide higher negative log perplexity than Google’s lexicon.

Effectiveness of Our Identified Critical Cases

In this section, we want to establish that the improved performance of our lexicon comes not only from the increase in the number of training samples but also due to the fact that the critical cases identified by our novel methodology have been added as training samples. For this, we

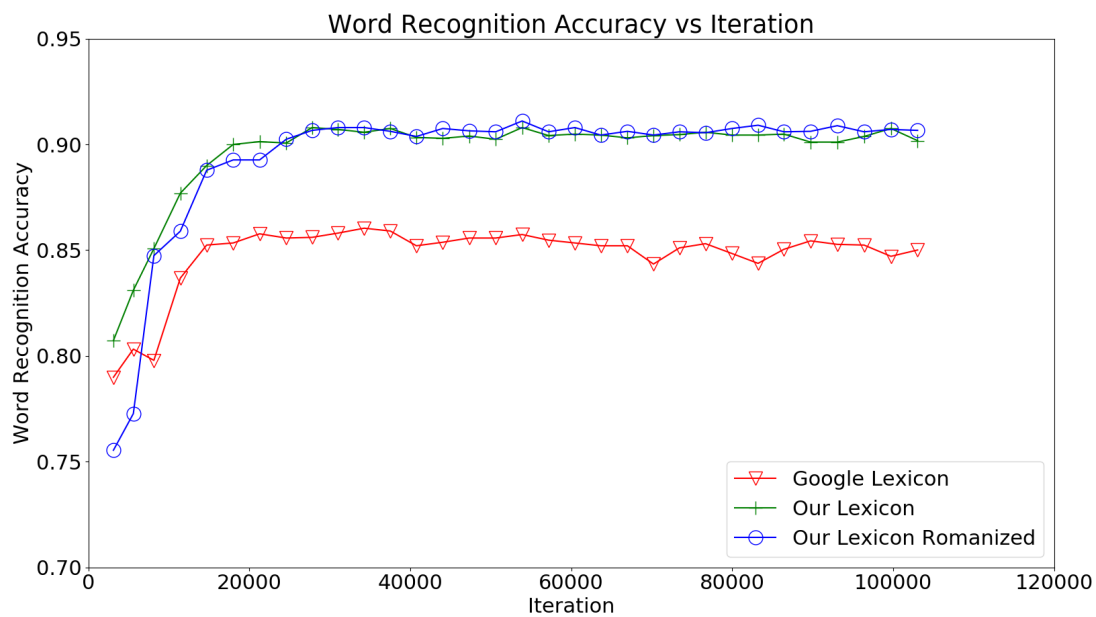


Figure 3.4: Word Recognition Accuracy vs Iteration

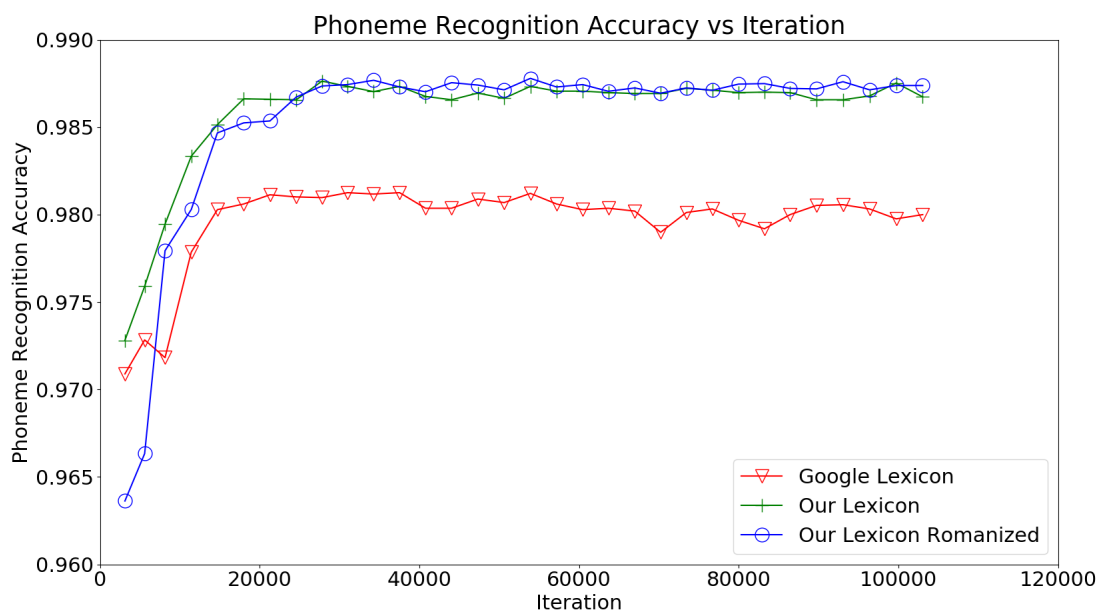


Figure 3.5: Phoneme Recognition Accuracy vs Iteration

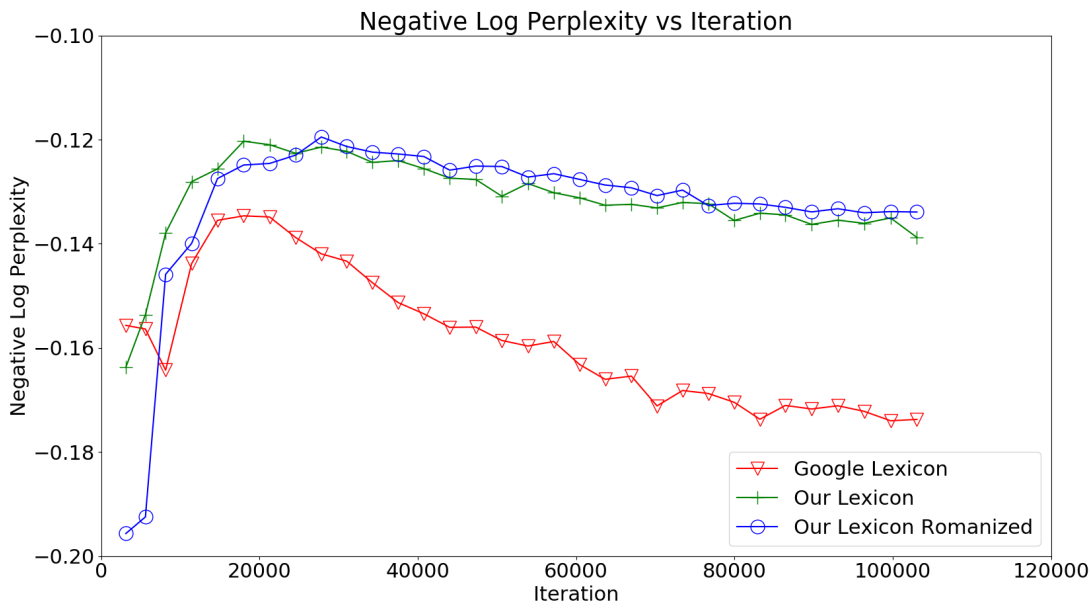


Figure 3.6: Negative Log Perplexity vs Iteration

Lexicon	Model	WER (%)	PER (%)
Google	LSTM-RNN	17.1	2.32
	Transformer Model	14.8	1.88
New Lexicon	LSTM-RNN	12.6	1.54
	Transformer Model	11.2	1.49

Table 3.6: Effectiveness of critical cases. Both lexicons are of size 60K. New Lexicon consists of 21K critical cases and 39K entries from Google lexicon.

prepare a new training lexicon by combining a portion of the Google lexicon with a portion of our identified critical cases. As we have kept 9K entries from the critical cases as our test set, we take the remaining 21K critical cases and combine them with the randomly taken 39K entries from Google lexicon to prepare a new lexicon of size 60K. While taking entries from the Google lexicon, we ensure that we do not take any repeated entry that has already been in the critical cases and added to the new lexicon. We then compare the performance of this new lexicon with the Google lexicon, both of which are of the same size (60K), on our test set. The results are in Table 3.6. The results clearly show that even in the case of same sized lexicons, our identified critical cases can significantly improve the performance as evidenced by the lower WER and lower PER than those for the Google lexicon.

Chapter 4

Speech Corpus Preparation

In this chapter, we describe our speech corpus development process. Specifically, we describe our domain study, speech corpus development approach in a supervised environment, automatic speech transcription approach, and synthetic speech corpus generation approach.

4.1 Voice Command Domain Study

Our primary objective was to cover all the voice assistant accessories that Bixby supports. Moreover, we consider the future scopes and select every possible domain in which we can recognize Bangla voice commands. We study the popular voice assistants - Google Assistant, Google Home, Amazon Alexa, Siri, Bixby, Cortana, and more. We explore them and collect sentences from Smart-phone commands (System commands, Contacts, Media Player, Camera, Gallery, Messaging, Weather, Date, Alarm, Email, etc.), Home appliances (Smart TV, Fridges, Air-Conditioners, Computers, etc.), Office work accessories (Projectors, Printers) and Automotive navigation applications (Vehicle routing, Utility Control). For automotive, we consider all voice commands supported by popular smart cars such as Ford Sync, Lexus Voice Command, Chrysler UConnect, Honda Accord, and GM IntelliLink. We prepare a list of 1700 voice commands that covers the entire target domain. Table 4.1, 4.2, 4.3 show summary of our voice command domain.

Domain Info	Number of Commands
Smartphone Navigation/Operation	340
Popular Apps	190
Home Appliances	440
Office	180
Automotive	550
Total	1700

Table 4.1: Target Domain of Voice Command

Included Smartphone Actions	
Launch Apps	basic commands
Call	Texting
Camera	Music
Set Alarm/ Reminder	Browse
Write Notes	Use Radio
Use Bluetooth	Wi-Fi

Table 4.2: Smartphone Operations Included

Included Apps	
Facebook	Twitter
Calendar	Weather
News apps	Pandora
Music apps	Navigation/Maps
Uber	Google Play Store
YouTube	Spotify
Sound Cloud	Tuneln

Table 4.3: Apps Included

4.2 Previously Available Speech Corpus

We consider all publicly available Bangla speech corpus as well as prepared a speech corpus of our own. The largest publicly available speech corpus is provided by Google [38]. Table 4.4 shows the summary of this corpus. It contains 217902 utterances from 505 speakers. Among the speakers, 323 of them are men and 182 women. The size of the speech corpus is approximately 220 hours.

4.3 Supervised Speech Corpus Development

Different approaches for speech corpus development have been explored by the researchers. A basic approach is to manually transcribe existing audio files. This is a very time consuming, monotonous, and error-prone task. Transcription of one-hour recording can take 3 to 5 hours or more [68]. A comparatively faster approach is to develop an interactive mobile application that

Aspect	Value
Number of Utterance	217902
Number of Speaker	505
Male Speaker	323
Female Speaker	182
Corpus Size	220 Hours

Table 4.4: Google’s Crowd-Sourced Speech Corpus

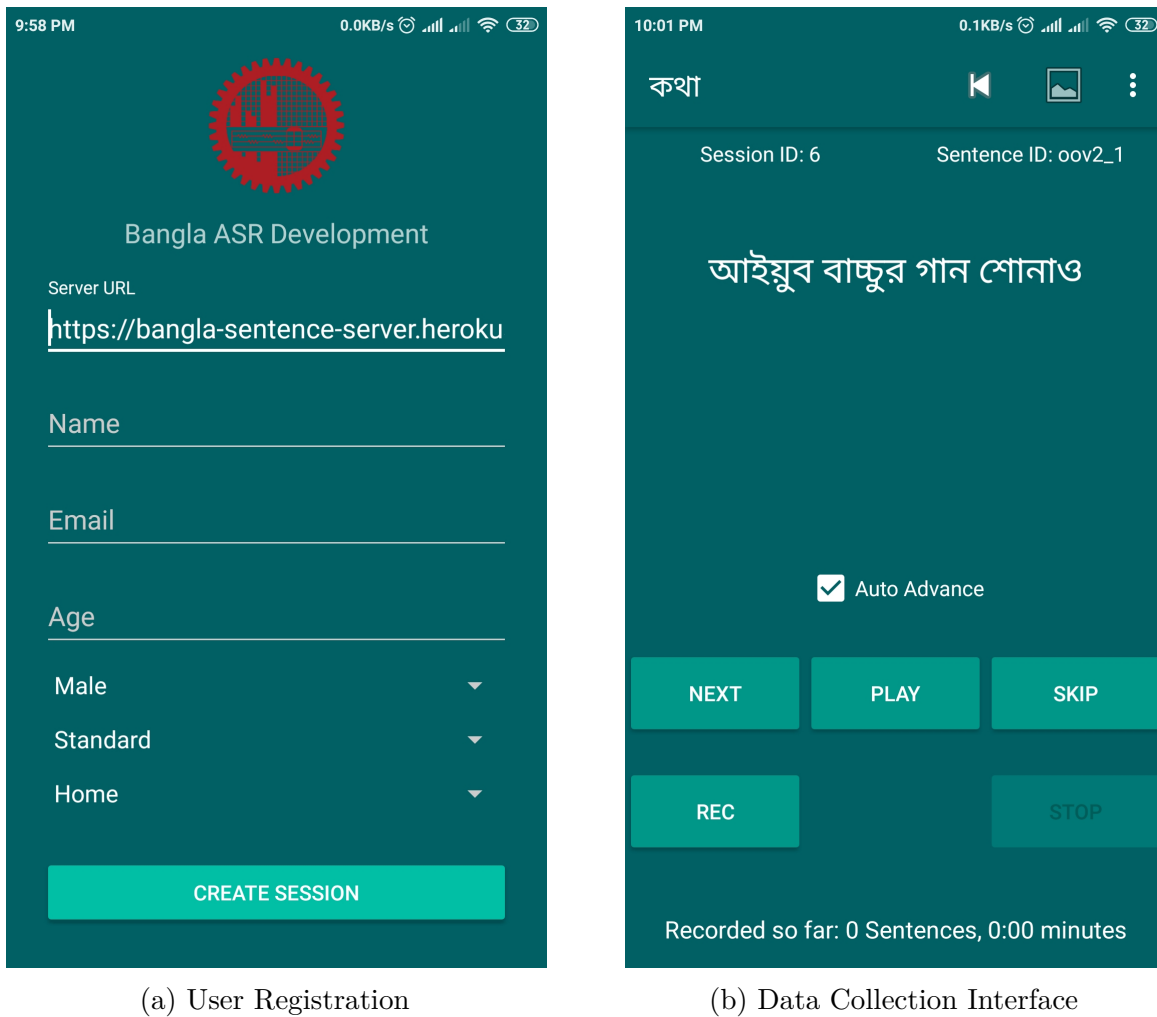


Figure 4.1: Data Collection App

prompts the user to read a particular text [39]. The user will have to start recording manually, read the prompted text, and end recording manually. In our experience, preparing one hour's worth of transcribed speech takes around 2 hours in this approach.

Bangla voice commands contain a set of technical words that are missing from all publicly available speech corpus. Also, the sentence structure of the voice commands is sometimes different from regular Bangla sentences. So we develop a speech corpus solely containing Bangla voice commands.

4.3.1 Data Collection App

We prepare an Android app for speech data collection following the approach by [39]. Figure 4.1 and 4.2 shows the app interface we used for speech data collection.

User Registration

During the user registration process, we collect the following information about the current recording session:

- User Name
- Email
- Gender
- Age
- Recording Environment
- Accent

After the registration process, the recording screen will appear.

Data Collection Interface

In the recording screen, the current sentence, current session-id, and current sentence id will be shown. The session id is unique to this session. The sentence id is unique to this text across all sessions. At the bottom of the recording screen the following buttons will appear:

At the bottom of the recording screen the following components will appear:

Start Button This button is used for starting the recording. This remains disabled if recording is running.

Stop Button This button is used for stopping the recording. This remains disabled if recording has not started yet.

Play Button This button is used for playing the audio file related to the current sentence if it had been recorded.

Next Button Save current recording and move to next sentence. A recorded sentence can be found, played, and updated at any time by finding it through the gallery.

Skip Button If the user is unable to understand the current sentence, he/she can choose to skip the sentence. A skipped sentence can be recorded anytime by finding it through the gallery.

Auto Advance Checkbox When the user becomes comfortable with fast recording, the auto-advance mode can be enabled using the checkbox. In auto-advance mode, the next sentence will appear as soon as the user finishes recording the current sentence. In this mode, the user can record up to 500 sentences in 30 minutes.

In the navigation bar, some more feature is given. Such as:

Back To Previous If any manual mistake needs to be corrected during the auto-advance mode, this button needs to be used. This is faster than finding the sentence in the gallery.

Open Gallery Shows all sentences recorded or skipped. The recorded sentences appear in green, the skipped sentences appear in red.

Terminate Session Used for terminating current recording session and sending information to the server.

Gallery Interface

Gallery interface shows all sentences recorded or skipped. The recorded sentences appear in green, the skipped sentences appear in red. Users can select any sentence to open the corresponding recording screen. Users can play recorded audio or re-record audio if necessary.

Data Upload

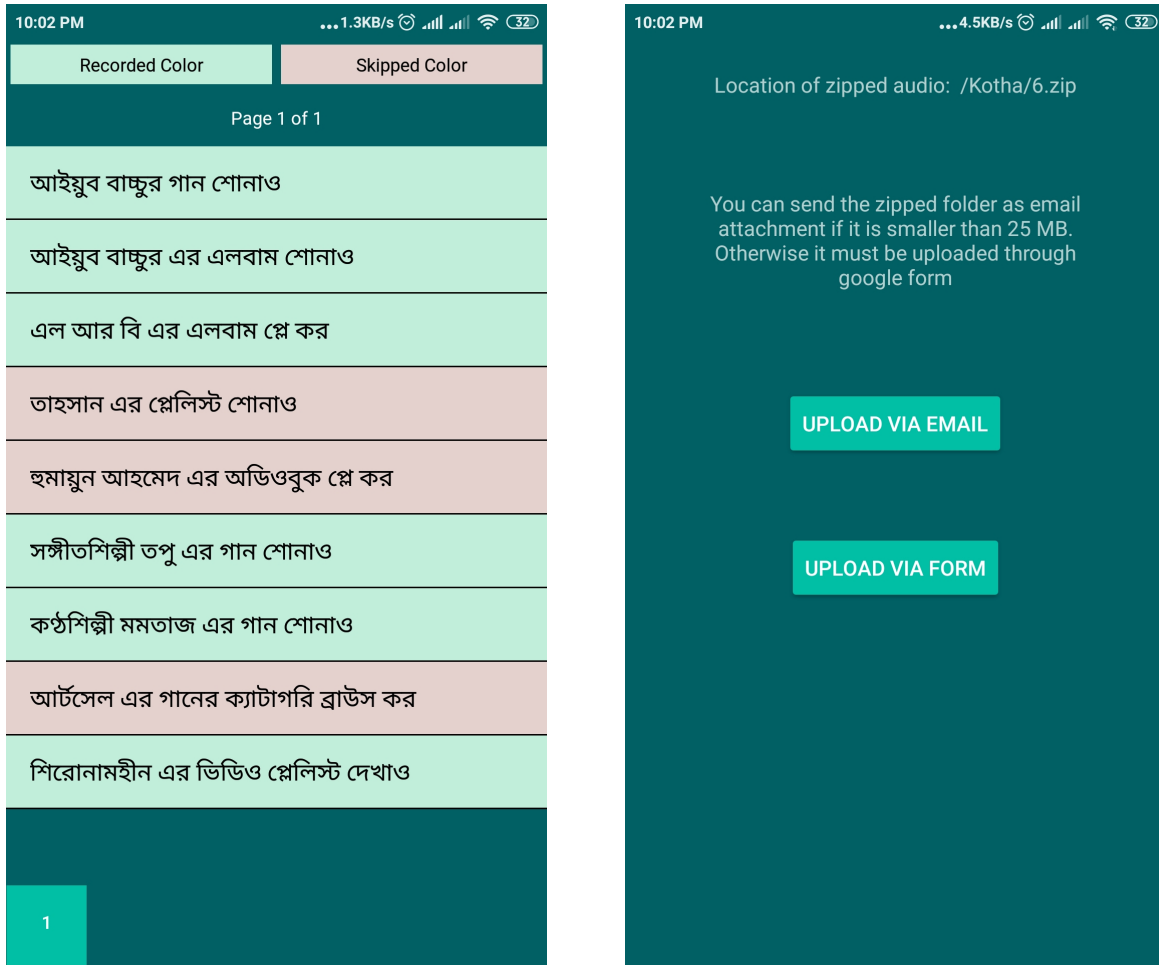
The recorded audio files are named in a particular format (session_id-sentence_id.wav). This makes it easy to find the corresponding text transcription of the audio and related speaker information. The recorded audio is 16kHz mono-channel audio files. After the user terminates the session, all recorded audio files will be zipped together. The user can then use the data upload interface to upload the zipped audio via email or a Google form. The email is auto-generated and the user only needs to select the sending email address and click send.

4.3.2 Summary of Voice Command Specific Corpus

Table 4.5 shows the summary of the corpus. We are able to collect 28973 sentences from 56 speakers using this application. Among the speakers, 34 are men and 22 are women. The size of this corpus is around 50 hours. 20 hours of speech data came from university student volunteers who participated in our workshop. Around 30 hours of speech data were collected by contacting NGOs who arranged data collection sessions among other volunteers. The age range of the speakers ranges from 20 to 35 with average age around 26.

4.4 Corpus Generation Using Automated Transcription

Figure 4.3 shows an overview of our system. We use publicly available audiobooks and TV news recordings collected from YouTube as an audio source in our system. All our audio files



(a) Speech Gallery

(b) Data Upload

Figure 4.2: Data Collection App

Aspect	Value
Number of Utterance	28973
Number of Speaker	56
Male Speaker	34
Female Speaker	22
Speaker Age (Avg)	26
Corpus Size	50 Hours

Table 4.5: Voice Command Corpus

4.4.2 Speaker Diarization

Speaker diarization refers to the task of grouping speech segments in an audio stream containing multiple speakers in a way that speech segments from the same speaker form a cluster. We follow the approach described in [71] for speaker diarization. This speaker diarization system (see also [72], [73]) is based on the binary key speaker modelling [74].

Binary key speaker modeling provides a compact and efficient representation of speech segments or clusters in the form of a vector. The vector captures speaker-specific features. The classification task is carried out by computing the similarity measures between binary keys. The proposed system obtained a Diarization Error Rate (DER) of 11.93%.

In our system, ICMC (Q transform Mel-frequency cepstral coefficients) were used in place of baseline MFCC acoustic features. For clustering, an affinity matrix is calculated from the data points. The eigenvectors corresponding to the top eigenvalues estimated from the affinity matrix is used as the similarity measure between data points. Then data points are clustered according to this similarity measure. Then we smooth and denoise the data, perform eigenvalue decomposition and sort the eigenvalues in descending order. Then we select the number of clusters according to the value which maximizes the eigengap. The spectral clustering algorithm often results in the estimation of a single speaker, therefore the system is configured to force the return of two or more clusters. Then the system performs pre-clustered thresholding of the eigengap between the two largest eigenvalues.

4.4.3 Gender Detection

We extract Mel Frequency Cepstrum Coefficients (MFCCs) features from the audio files. A lot of acoustic features like peak frequency (the frequency with the highest energy), meanfun (average of fundamental frequency measured across acoustic signal), minfun (minimum fundamental frequency measured across acoustic signal), etc. are included in MFCC features.

We use a Gaussian Mixture Model to build the gender detection system from these extracted features. The training dataset consists of Mozilla common voice data set. We had almost 58,000 male voice clips and 17,000 female voice clips in the train set. After training, a test data set consisting of manually tagged Bangla audio clips are used for evaluation. The test set had 826 male and 590 female audio clips. Even though the training set and test set had completely different languages, we achieved a recognition rate of 85% and 98% for male and female clips respectively.

4.4.4 Silence Based Segmentation

We segment each of the audio files on silence intervals and ensure that all the audio segments are less than or equal to 35 seconds. We use PyAudioAnalysis [75] for this task.

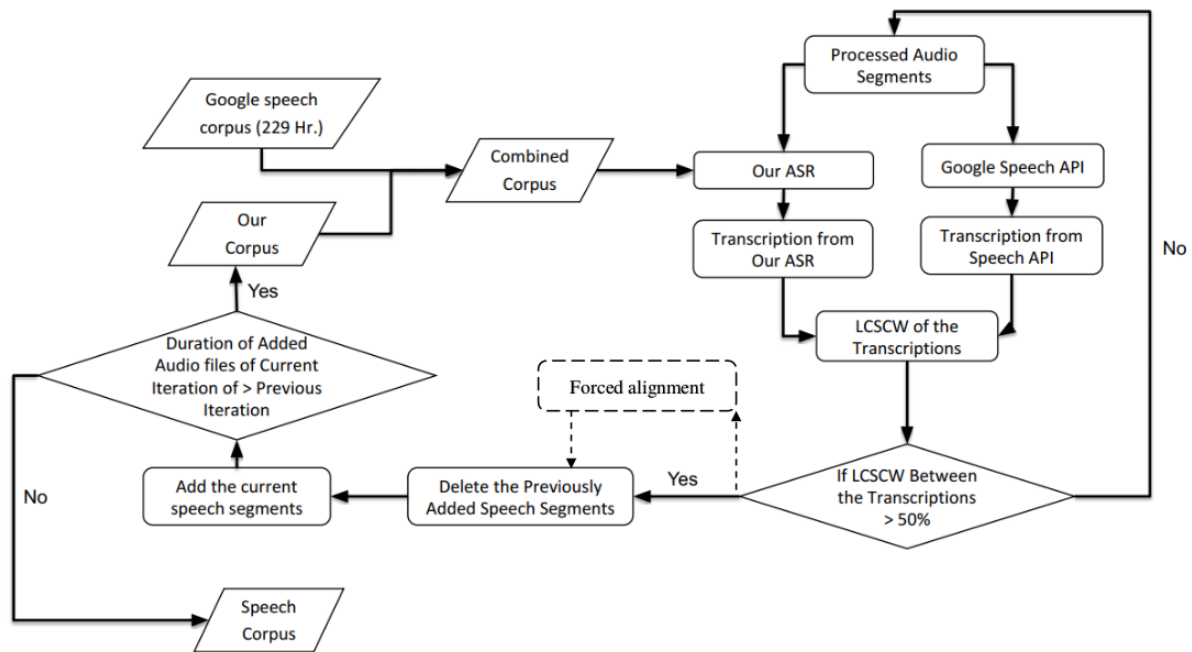


Figure 4.4: Overview of Automatic Transcription

We take 0.4 seconds as minimum silence length (the minimum length of the silence at which a split may occur) and 0.0001 as silence threshold (the energy level (between 0.0 and 1.0) below which the signal is regarded as silent).

4.4.5 Automatic Transcription Generation

As we did not have any reference text for any of the audio files, we had to automatically generate the transcriptions. We designed and implemented an iterative algorithm for this task (Algorithm 5). This algorithm automatically transcribed processed speech data, does sanity testing, and adds the accurate speech-text segments to the speech corpus.

This algorithm uses two speech recognition systems: one is Google Speech API and another is our speech recognition system that has been trained on publicly available 220 hours of speech data from Google (out of the remaining 3 hours, 2 hours for the test set, 1 hour for validation set). We use a hybrid CTC-Attention based end to end system for training our ASR [76]. The flowchart of the automatic transcription process is given on 4.4

We observe that none of the two systems provide fully accurate performance and Google API provides much better performance than our system. To generate transcription with reasonable confidence, we decide to generate transcriptions using both ASR models. Our intuition is that, for each of the audio files, if we take the longest common sequence of consecutive words between the outputs of both the systems and take only the audio and transcription for that matched portion, we can be confident enough about the accuracy of the transcription. However, we cannot do this at the start because our speech recognition system was performing quite

poorly initially. So, we follow an iterative strategy where we add the Google Speech API transcription to our speech corpus if it passes the sanity test performed by our developed ASR system. To perform the sanity test, we generate the transcriptions of the audio files from both of the systems. We consider the longest common sequence of consecutive words from both of the transcriptions. We take the percentage of the length of this matched portion with respect to the length of transcription from Google API. If this percentage value is greater than a threshold (50%), then the speech API transcription has passed the sanity test. Even though Google API transcription passing the sanity test still has some errors, adding these transcriptions to our corpus increases our ASR performance due to the larger corpus size. In the second iteration, we use our ASR trained on the extended speech corpus to perform sanity testing of the Google speech API transcription again. Because in the first iteration, some data may have failed the sanity testing because of the errors introduced by our own ASR. In the second iteration, more data passes the sanity test and we add them to our speech corpus.

At each iteration, we increase the number of training data to get a better model in the next iteration. Note that, we only try to increase the performance of our speech recognition system. We stop iterating when the number of newly added training samples does not increase much compared to the previous iteration. At the start of the last iteration, we delete the training samples added at the previous iterations. Finally, for each of the audio files, we compare our transcription with Google API transcription and we take the longest common sequence of consecutive words between the outputs of the two models. We take only the audio and transcription for that matched portion to be included in our final speech corpus. Thus, the final corpus becomes free from any transcription error introduced by either ASR model. We only consider exact matching within our threshold. The matching threshold of 50% is intuitive. It may be possible to improve the algorithm by tuning this threshold. But we avoid doing that due to the computational complexity of the iterative corpus generation algorithm.

In this approach, we exploited the performance gap between our ASR and Google Speech API to extend the speech corpus. We refer to the corpus generated from Automatic transcription as 'Transcribed corpus'. The size of the Transcribed corpus is around 510 hours.

4.4.6 Evaluation of Automatic Transcription

Figure 4.5 shows the histogram of the percentage of the longest common sequence of consecutive words (LCSCW). We calculate it in the following way. We calculate the LCSCW between the transcription provided by our ASR and Google Speech API. We calculate the percentage of LCSCW with respect to the transcription length provided by the Google Speech API. We plot the histogram of these percentages within 10 ranges: 0-10%, 10-20%, etc. Each color in the graph represents a particular iteration. We can see from figure 4.5 that in the earlier iterations, most of the LCSCW percentages are in shorter regions. Iteration 1 has the

Algorithm 5 Iterative Algorithm for Transcription

```

1:  $ot \leftarrow$  Our ASR transcription
2:  $gt \leftarrow$  Google API transcription
3:  $lcscw \leftarrow$  Longest common sequence of consecutive words
4:  $gc \leftarrow$  Google speech corpus
5:  $oc \leftarrow$  Our speech corpus
6:  $dc \leftarrow$  Duration of train data at current step
7:  $dp \leftarrow$  Duration of train data at previous step
8:  $d\delta \leftarrow$  Change in duration of speech corpus
9:  $al \leftarrow$  List of audio files
10: while  $d\delta \neq 0$  do
11:   Train ASR on  $(gc + oc)$ 
12:    $oc \leftarrow \{InitialCorpus\}$ 
13:    $dc \leftarrow 0$ 
14:   for each  $audio$  in  $al$  do
15:     generate  $ot$ 
16:     get  $gt$ 
17:      $lcscw \leftarrow lcscw(ot, gt)$ 
18:      $percentage \leftarrow \frac{len(lcscw)*100}{len(gt)}$ 
19:     if  $lcscw\_percentage > 50\%$  then
20:        $dc \leftarrow dc + audio\ duration$ 
21:        $oc \leftarrow oc + gt$ 
22:      $d\delta \leftarrow dc - dp$ 
23:      $dp \leftarrow dc$ 
24: Train ASR on  $(gc + oc)$ 
25:  $oc \leftarrow \{InitialCorpus\}$ 
26: for each  $audio$  in  $al$  do
27:   generate  $ot$ 
28:   get  $gt$ 
29:    $lcscw \leftarrow lcscw(ot, gt)$ 
30:    $percentage \leftarrow \frac{len(lcscw)*100}{len(gt)}$ 
31:   if  $lcscw\_percentage > 50\%$  then
32:      $oc \leftarrow oc + matched\ audio\ segments$ 
33: return  $oc$ 

```

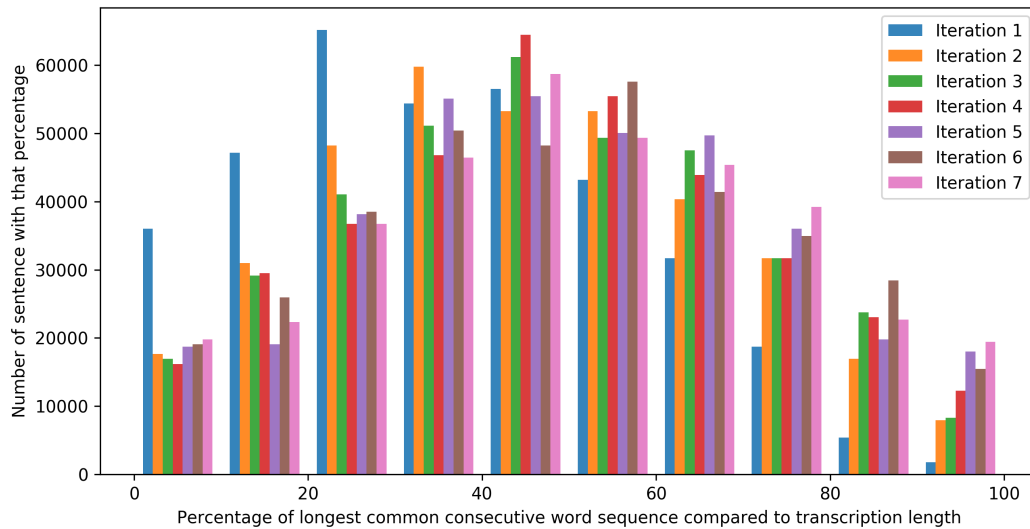


Figure 4.5: Histogram for percentage of longest common consecutive word sequence length between two transcriptions with respect to transcription from Google Speech API

Iteration	Transcribed Corpus size	WER (%)
1	207 hours	25.98
2	379 hours	24.25
3	426 hours	23.64
4	464 hours	23.46
5	492 hours	23.22
6	509 hours	23.08
7	512 hours	23.00
Forced alignment	507 hours	22.70

Table 4.6: Evaluation of Corpus by Iteration

highest frequency in the lower percentage area. As we add more data to the training corpus, the performance of our ASR increases. It starts recognizing more words accurately, resulting in a longer common consecutive word sequence length. We can see that in iteration 6 and 7, there are more sentences with higher LCSCW percentages. Figure 4.5 shows the rightward shift of the histogram during different iterations of algorithm 5.

Table 4.6 shows the evaluation of the corpus generated at each iteration of our algorithm. The second column shows how many transcribed speech data were generated at that particular iteration. We train a hybrid CTC-Attention based end to end system using each corpus and evaluate the performance of that system. We use our generated corpus at each step in addition to the Google speech dataset for evaluation. We can also see in table 4.6 that the amount of transcribed corpus generated at each iteration and corresponding WER reaches saturation after only 6-7 iterations. One possible reason is the limited ability of the system to add variance to

Aspect	Value
Number of Utterance	150,000
Number of Speaker	5190
Male Speaker	2680
Female Speaker	2510
Speaker Age (Avg)	40
Corpus Size	510 Hours

Table 4.7: Automatically Transcribed Corpus

the existing training corpus. The size of the corpus after the system reaches saturation is around 510 hours. The drawback of this system is that it fails to utilize all collected audio files. But there are two key benefits. It allowed us to transcribe 510 hours of speech data very quickly. Also, this system is very useful in cases where the forced alignment technique cannot be used directly (i.e., no reference text available).

4.4.7 Summary of Transcribed Corpus

Table 4.7 shows the summary of the corpus. We are able to collect around 150K sentences from 5190 speakers using this application. Among the speakers, around 2680 are men and 2510 are women. The size of this corpus is around 510 hours. Due to the crowd-sourced nature of the speech corpus, it is difficult to estimate the age distribution of the speakers. Since most of the transcribed speech corpus is from news anchors of popular Bangla TV channels, we tried to estimate an age distribution from the age information of known TV anchors. The average age for male and female TV anchors is 41.5 and 39.3 respectively. The age of the anchors ranges from 22 to 60.

4.5 Synthetic Speech Generation for OOV words

In this section, we describe our approach for synthetic corpus generation for out-of-vocabulary Bangla words.

4.5.1 Out-of-Vocabulary Word List

We first prepare a large Bangla text corpus. Our text corpus has 10 million Bangla sentences containing 1.7 million unique words. Among these words, 56000 words occur at least once in the speech corpus. The rest of the words are considered out-of-vocabulary.

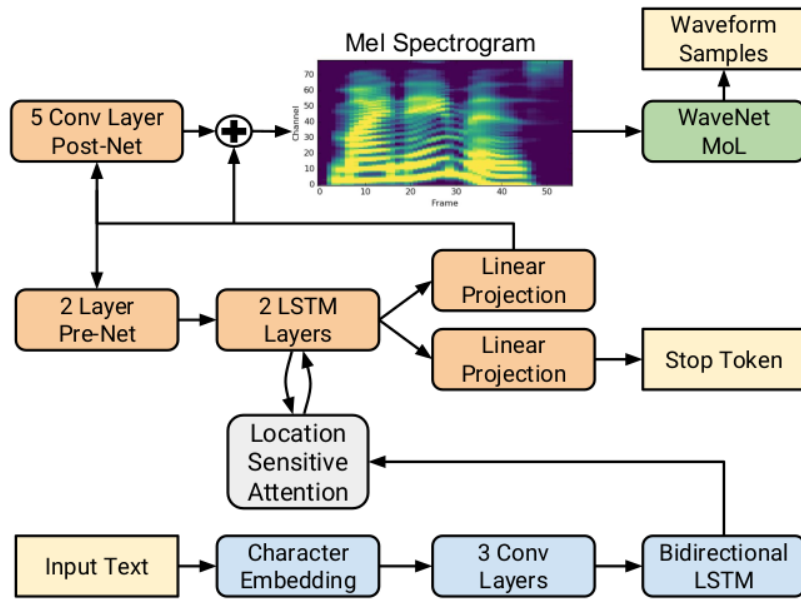


Figure 4.6: Overview of our Text-to-Speech Architecture

4.5.2 TTS Model

We prepare our text-to-speech (TTS) system using ESPnet-TTS [77]. Specifically, we use the Tacotron 2 [78] implementation of ESPnet-TTS. Figure 4.6 shows the Tacotron 2 TTS architecture. Tacotron 2 is a Recurrent Neural Network (RNN) based sequence-to-sequence network. It has a bi-directional LSTM based (BLSTM) encoder and a unidirectional LSTM-based decoder. Additionally, it uses a location-sensitive attention mechanism. In our implementation, the encoder network has 1 layer with 512 BLSTM units. The decoder network has 2 layers with 1024 unidirectional LSTM units in each layer.

4.5.3 Speech Synthesis

Speech synthesis is done in the following manner. First, our TTS model takes an input text sequence and generates log Mel filter bank feature sequence. Then log Mel filter bank feature sequence is converted to a linear spectrogram. Finally, the Griffin-Lim algorithm [79] is applied to the spectrogram to generate audio.

We use out-of-vocabulary words as input for our text-to-speech system. We refer to the corpus generated from Speech synthesis as 'Synthesized Corpus'. The size of the Synthesized corpus is around 450 hours.

Aspect	Google	Voice Command	Transcribed	Synthesized	Total
Num of Utt	217902	28973	150,000	1.64M	2M
Num of Speaker	505	56	5190	1	5752
Male Speaker	323	34	2680	N/A	3037
Female Speaker	182	22	2510	N/A	2714
Speaker Age (Avg)	N/A	26	40	N/A	38.75
Size (Hours)	220	50	510	450	1230

Table 4.8: Overall Speech Corpus

4.6 Speech Corpus Summary

Table 4.8 shows the summary of our entire speech corpus. The overall size of the corpus is around 1230 hours and the number of speakers is 5752.

Chapter 5

Speech Recognition Architecture

In this chapter we describe the speech recognition architectures we used for our work. We experiment on both traditional and end-to-end architectures.

5.1 Traditional ASR System

For our traditional ASR, we use the traditional Hidden Markov Model (HMM) based ASR recipe provided by Kaldi ASR engine. It has three main components: the acoustic model, language model and the phonetic dictionary. We use Mel Frequency Cepstral Coefficients (MFCC) as our speech feature. Figure 5.1 shows the overview of a traditional ASR system.

5.1.1 Speech Feature Extraction

During feature extraction, each audio is split into a frame of 25 millisecond. Then MFCC feature is extracted from each audio frame. Mel-frequency cepstral coefficients (MFCCs) are coefficients are derived from a type of cepstral representation of the audio clip. In this representation, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum. MFCCs are commonly derived from an audio frame as follows:

- Taking the Fourier transform of the speech audio frame. Audio frame duration is 25 millisecond in our case.
- Mapping the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows. The overlapping window duration is 10 millisecond for our work.
- Taking the logs of the powers at each of the mel frequencies.
- Taking the discrete cosine transform of the list of mel log powers.

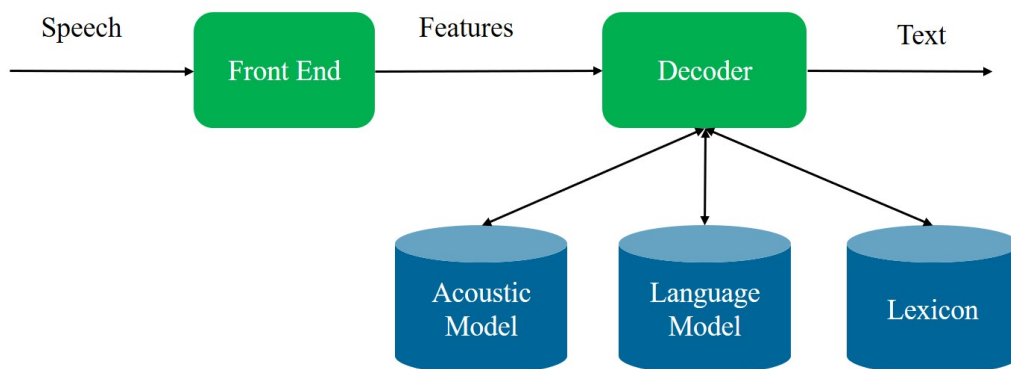


Figure 5.1: Traditional ASR Overview

- The MFCCs are the amplitudes of the resulting spectrum.

5.1.2 Acoustic Model

The acoustic model provides a probability for a speech feature to originate from different phonemes or sub-phonemes. We use Gaussian Mixture Model (GMM) as our acoustic model. We have 47 phonemes in our phoneme list. Each phoneme is composed of three sub-phonemes. So there are 141 sub-phonemes in total. For a particular speech feature, GMM provides the probability of this feature to originate from each of the 141 sub-phonemes.

5.1.3 Language Model

The language model provides the probability of a word of a sequence given other nearby words. It can be very useful when the pronunciation is not very clear or when there are multiple candidate words sounding similar. The language model improves the ASR performance by providing some contextual relevance of a candidate word and removes ambiguities. We use trigram based language model in our work. It is separately trained on our Bangla text corpus mentioned in section 3.2.

5.1.4 Phonetic Dictionary

The phonetic dictionary provides the maps the phonetic transcription of word to its written representation. We use the phonetic dictionary described in section 3.3. It has phonetic transcription for around 100K most frequent Bangla words.

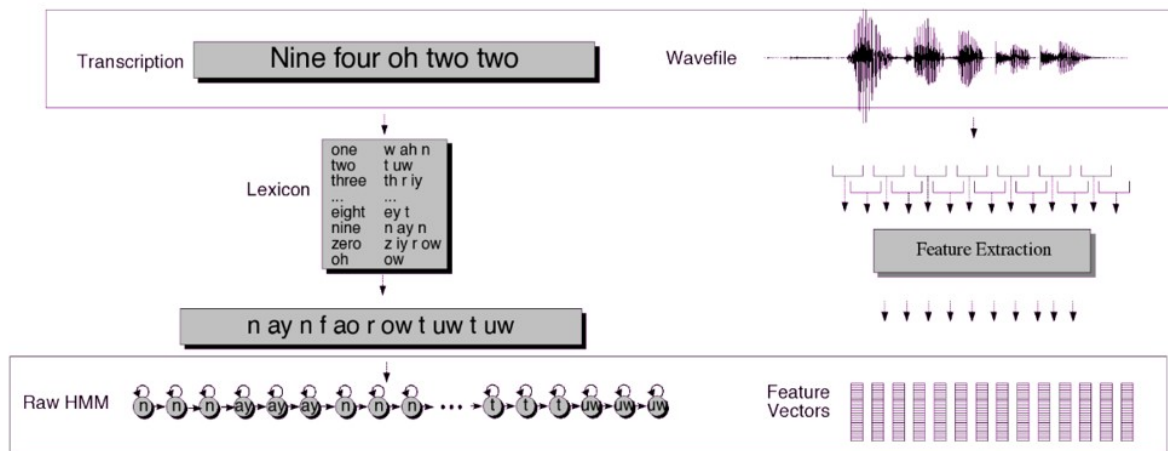


Figure 5.2: Traditional ASR Training

5.1.5 Training Phase

Figure 5.2 shows the training phase of the ASR system. A text-speech pair is taken by the system. Each text is a sequence of words. Each word is replaced by its corresponding phonetic transcription in the lexicon. Each phoneme is then replaced by three corresponding sub-phonemes. On the other hand, MFCC feature sequence is extracted from the speech. Then sub-phoneme sequence and MFCC feature sequence is aligned together. This alignment information is used by the GMM model that learns predict the underlying sub-phoneme probabilities by looking at the MFCC feature. Sub-phoneme to sub-phoneme transition probabilities are also learned and used by the HMM as state transition probabilities.

5.1.6 Decoding Phase

Figure 5.3 shows the decoding phase of the ASR system. It is based on Hidden Markov Model (HMM). Hidden Markov Model is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobservable hidden states. HMM assumes that there is another observable process whose behavior depends on the hidden process. The goal is to learn about the hidden process by observing the observable process.

In the context of speech recognition, the observable states are MFCC feature sequence. The hidden states are the sub-phonemes. The probability of a hidden state (sub-phoneme), given the observable state (MFCC feature) is modeled by Gaussian Mixture Model. This model as well as the hidden State transition probabilities (sub-phoneme to sub-phoneme) is learned during the training phase as described in the previous section. These two probabilities along with the language model probabilities are used by the viterbi decoder during decoding phase.

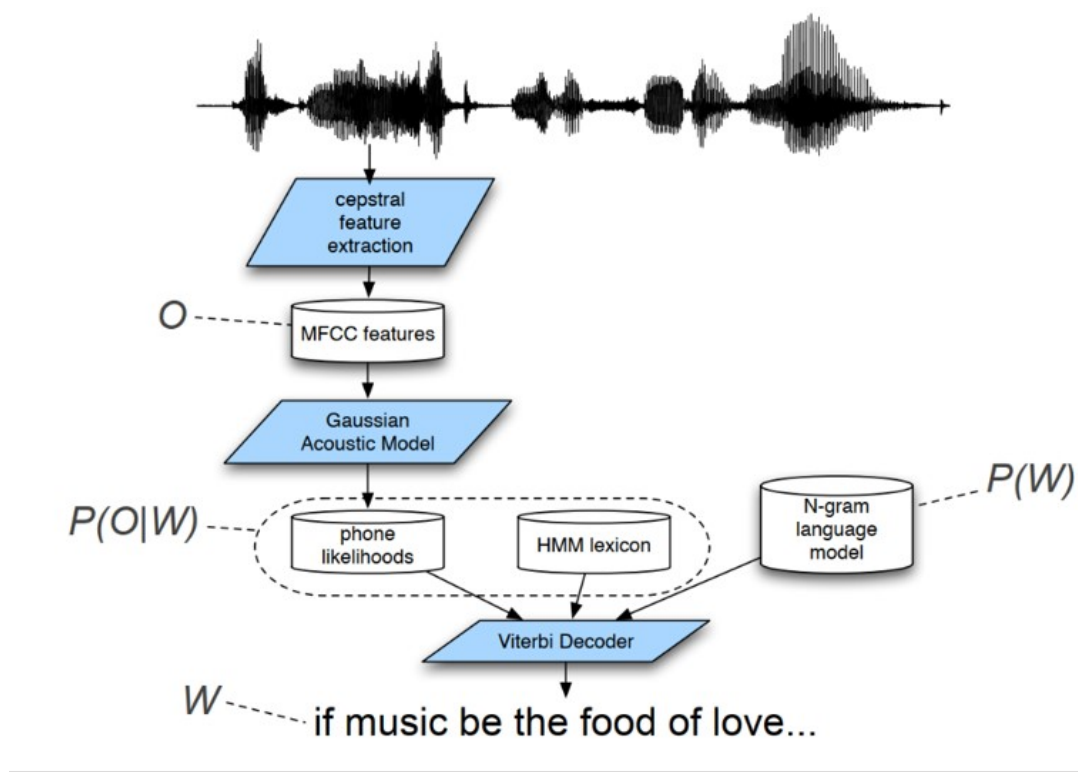


Figure 5.3: Traditional ASR Decoding

5.2 End-to-End System

The end-to-end system does not use any phonetic representation at an intermediate step. It directly tries to predict the grapheme sequence from speech feature sequence. Therefore, a phonetic dictionary is not required by the end-to-end models. Figure 5.4 shows the overview of our end-to-end ASR. It has an encoder unit with Bi-directional Long Short Term Memory (BLSTM) Units, a Connectionist Temporal Classification (CTC) unit and an attention based decoder. We also incorporate a Recurrent Neural Network (RNN) based language model with this system using shallow fusion.

5.2.1 Speech Feature

For each audio frame, we use 40 MFCC features along with their first and second-order temporal derivatives. This gives us 120 features per frame. The size of audio frame is 25 millisecond. MFCC feature extraction is already described in section 5.1.1.

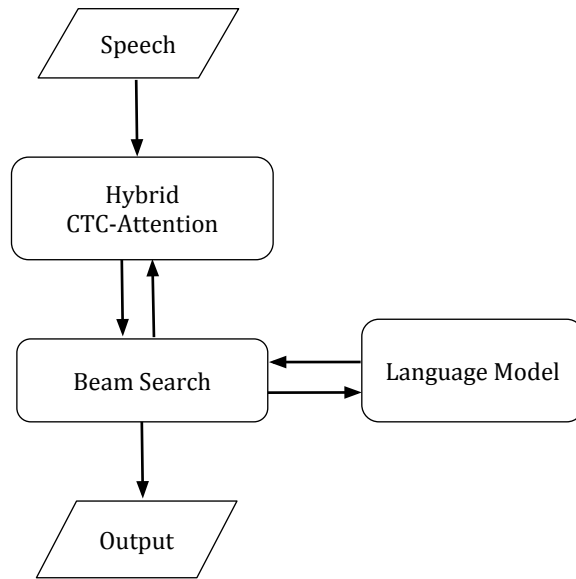


Figure 5.4: Overview of End-to-End Architecture

5.2.2 CTC-Attention

Our End-to-End architecture is based on the work of [76]. It is shown in Figure 5.5. We use hybrid of CTC and attention mechanism. CTC and attention encoder networks share the same Bidirectional Long Term Memory Units (BLSTM). The encoder network had 4 layers with 320 BLSTM cells in each layer. The linear project layer has 320 cells. It is followed by each BLSTM layer. The decoder network has 1 layer. It has 320 unidirectional LSTM cells. The shared encoder absorbs the input sequence into hidden states and the attention decoder generates the letter sequence. The CTC network also contributes in picking the best possible letter sequence by providing the align scores between speech features and letter sequence.

During decoding with beam search, both attention scores and CTC scores are combined in the following manner. Let $p(o_n)$ be the probability of output label o_n at position n , given previous out labels and w_1 be the CTC weight.

$$\log p^{hyb}(o_n) = w_1 \log p^{ctc}(o_n) + (1 - w_1) \log p^{att}(o_n) \quad (5.1)$$

Here CTC weight w_1 is a hyper-parameter that needs to be tuned.

5.2.3 Language Model

The language model is trained on a large Bangla Text corpus. We experiment with both word-level and character-level Recurrent Neural Network (RNN). For word-level RNN, we use 1 hidden layer with 1000 LSTM cells. Most frequent 65000 Bangla words are considered in our vocabulary. For character level RNN, we use 2 hidden layers with 650 LSTM cells each.

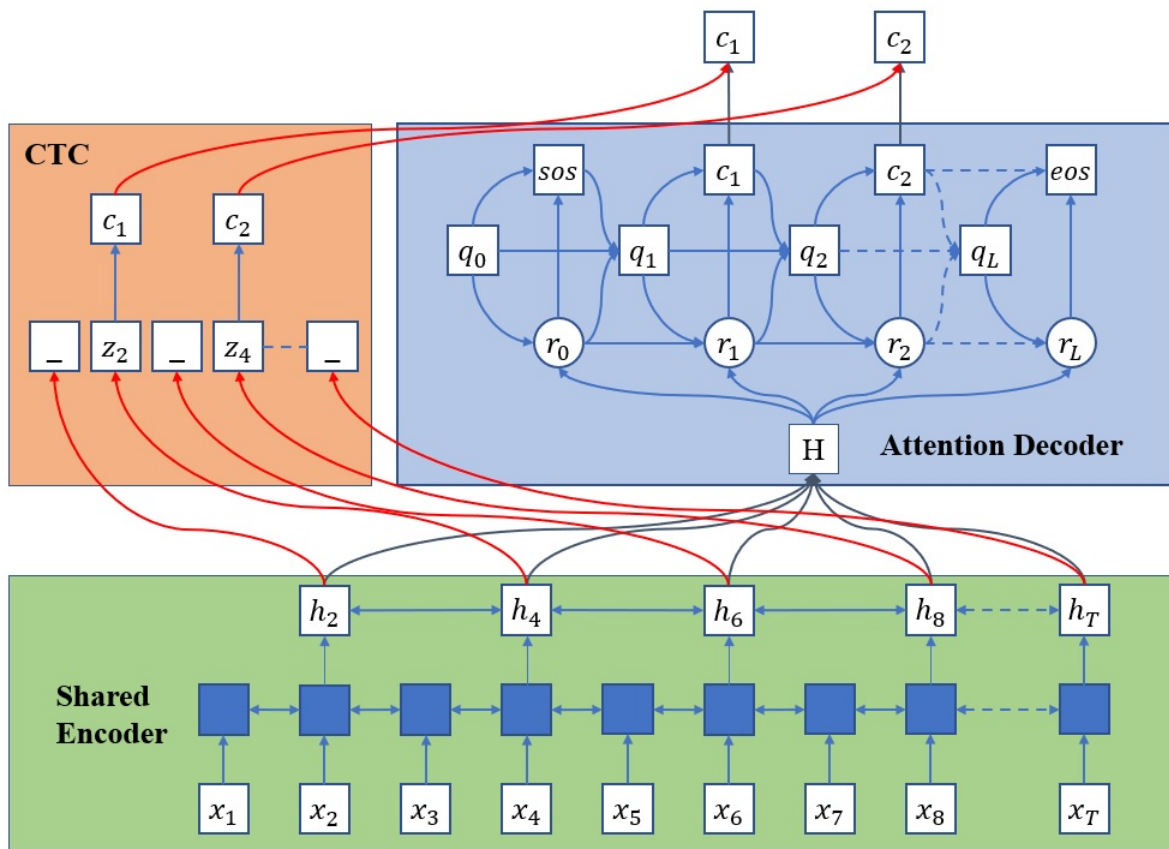


Figure 5.5: End-to-End Architecture

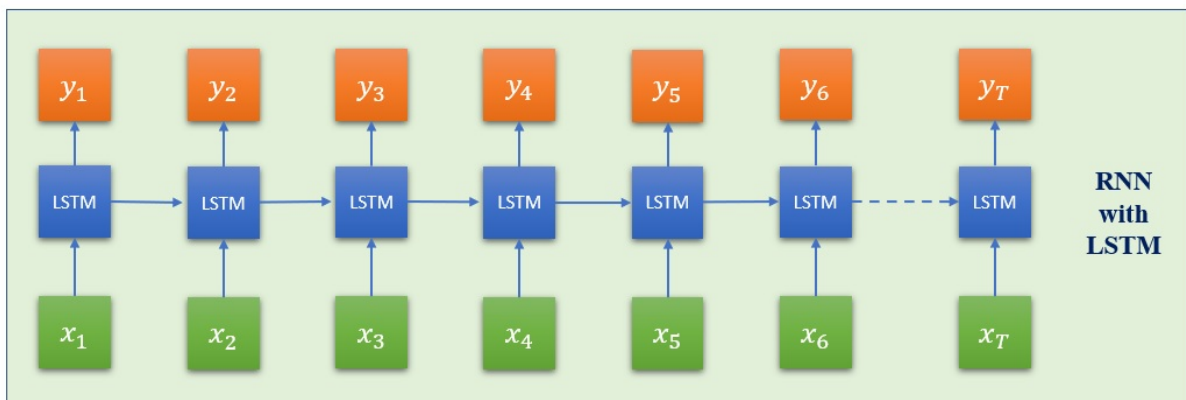


Figure 5.6: RNN Language Model

5.2.4 Beam Search

We use shallow fusion technique to combine the language model scores into the End-to-End system [80]. Let, b be the width of beam search and v be the vocabulary size. At each step of beam search, b partial hypotheses are maintained by the system. In the next beam search step, each of these b hypotheses is extended by each of the tokens in the vocabulary. The total number of candidates becomes bv . For each of these candidates, the following score is calculated.

$$\log p(o_n) = \log p^{hyb}(o_n) + w_2 \log p^{lm}(o_n) \quad (5.2)$$

Here, $p(o_n)$ be the probability of output label o_n given the previous output labels, $p^{hyb}(o_n)$ be the score from hybrid CTC-attention system, $p^{lm}(o_n)$ be the language model score and w_2 be the language model weight. After calculating scores for each of the bv candidates, the top b candidates are considered for the next beam search step. The language model weight w_2 is a hyper-parameter that needs to be tuned.

Chapter 6

Context Specific Optimization of Voice Commands

Voice command recognition task commonly involves an Automatic Speech Recognition (ASR) system with context-specific optimization. Context information for a specific smartphone user includes contact names, installed apps, songs, media files, location, recent search history, the content of the screen user is looking at, etc. Figure 6.1 shows the overview of context-specific optimization of ASR system. This context information changes frequently so it is desired that the contextual model will be updated on-the-fly within the device.

Some notable work on contextual speech recognition include [43], [44], [46], [47], [50], etc. Google has incorporated contextual information with their state-of-the-art speech recognition system [40], [42], [41] and more recently with End-to-End speech recognition system [45]. All of the approaches used by Google are variations of n-gram based model for context detection. We propose a multi-label topic modeling approach for context detection which has several advantages over the n-gram based approach. N-gram based approach is too rigid. It is not robust to synonymous, missing, or misplaced words. All possible synonyms and n-gram variations need to be present in the contextual corpus. The topic modeling approach works on keywords which is more flexible and robust than the n-gram approach. A variable number of contexts can be easily handled with multi-label topic modeling.

Our contribution in this work is the following.

- We propose multi-label topic modeling based contextual rescoring for Bangla Voice Command recognition
- We consider a wide range of Bangla voice commands (for smart-phone, home appliances, automobiles, etc.)
- Our rescoring system achieves WER of 12.8% when provided the context accurately. It outperforms all other existing voice command recognition systems in Bangla.

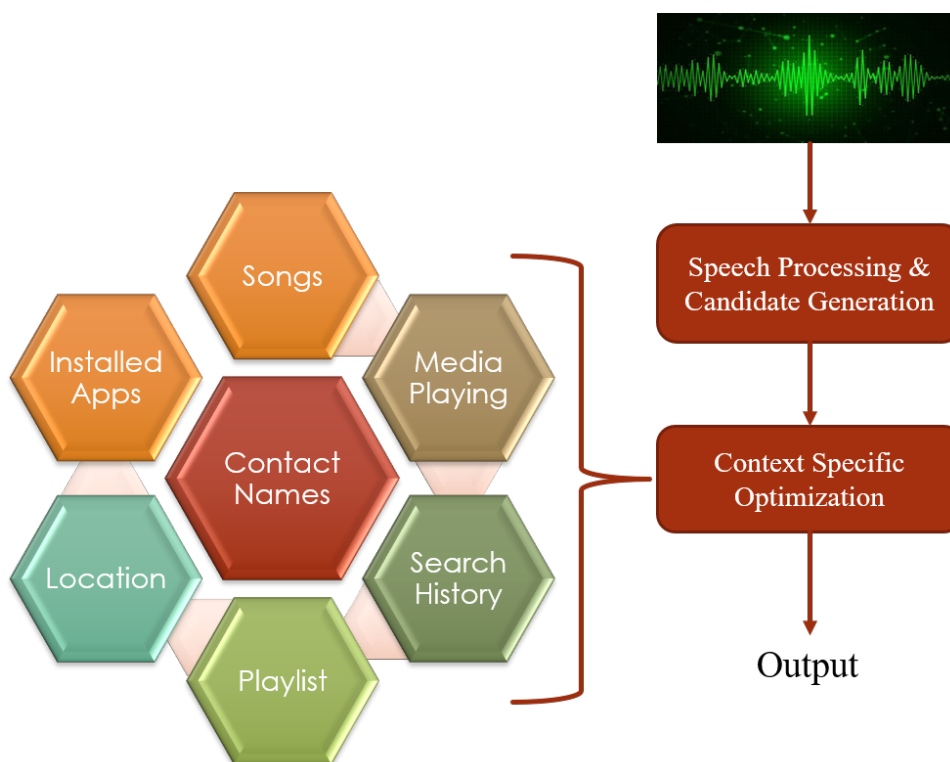


Figure 6.1: Contextual ASR Overview

6.1 Related Works

In various languages, researchers have used contextual information to increase the performance of voice recognition systems. [40] present an online approach for adjusting language model weights of n-grams corresponding to a specific context. [41] describe a composition based on-the-fly re-scoring mechanism to employ contextual language models in a speech recognition system. [42] use Named-Entity Recognition within the automatic speech recognition word lattice for identifying contextually related paths. They report that their approach minimizes Word Error Rate (WER) by 12.0% on a media playing commands data set. [43] provide a mechanism to learn contextual information in an unsupervised manner and for building automatically contextually biased models. [44] discuss two interpolation methods to merge contextual information with knowledge from a general language model. [45] consider contextual information during beam search in an end-to-end speech recognition system. [46] discuss contextual recurrent neural network language model. They consider a contextual input vector for each word of a sentence. [47] use class-based language models which provide contextual information during decoding in an end-to-end speech recognition system. Moreover, [48] addresses an end-to-end ASR for low-resource multilingual ASR context. [49] develop an end-to-end automatic speech recognition system that is situation informed. They consider speaker gender, conversational history, etc. to develop the situation-informed system. [50] develop an end-to-end speech recognition system

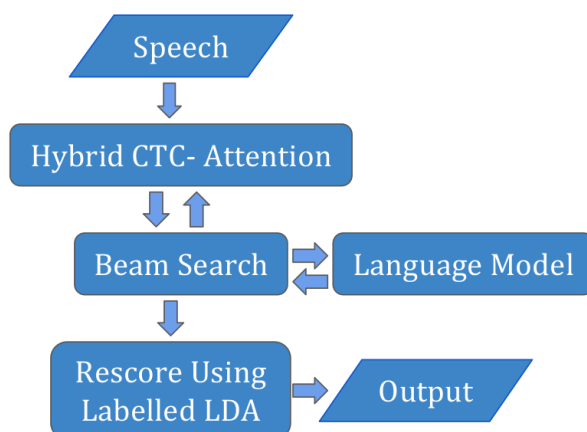


Figure 6.2: System Overview

that considers dialog context. We have not found any research work that focuses on considering contextual information for voice command recognition in Bangla language.

6.2 Our System

In this section, we describe our system in details.

6.2.1 System Overview

We use End-to-End ASR in our system. We use shallow fusion technique similar to the system described by [80] to incorporate the language model with the End-to-End architecture. Scores from CTC-Attention and language model are combined during beam search to generate a set of candidate hypotheses. Then we apply contextual rescoring on these candidates using Labeled LDA.

6.2.2 End-to-End Architecture

Our End-to-End architecture is based on the work of [76]. We use hybrid of CTC and attention mechanism. CTC and attention encoder networks share the same Bidirectional Long Term Memory Units (BLSTM). The encoder network had 4 layers with 320 BLSTM cells in each layer. The linear project layer has 320 cells. It is followed by each BLSTM layer. The decoder network has 1 layer. It has 320 unidirectional LSTM cells. For each audio frame, we use 40 MFCC features along with their first and second-order temporal derivatives. This gives us 120 features per frame. The shared encoder absorbs the input sequence into hidden states and the attention decoder generates the letter sequence.

During decoding with beam search, both attention scores and CTC scores are combined in the following manner. Let $p(o_n)$ be the probability of output label o_n at position n , given previous output labels and w_1 be the CTC weight.

$$\log p^{hyb}(o_n) = w_1 \log p^{ctc}(o_n) + (1 - w_1) \log p^{att}(o_n) \quad (6.1)$$

6.2.3 Language Model

The language model is trained on a large Bangla Text corpus. We experiment with both word-level and character-level Recurrent Neural Network (RNN). For word-level RNN, we use 1 hidden layer with 1000 LSTM cells. Most frequent 65000 Bangla words are considered in our vocabulary. For character level RNN, we use 2 hidden layers with 650 LSTM cells each.

6.2.4 Beam Search

We use shallow fusion technique to combine the language model scores into the End-to-End system [80]. Let, b be the width of beam search and v be the vocabulary size. At each step of beam search, b partial hypotheses are maintained by the system. In the next beam search step, each of these b hypotheses is extended by each of the tokens in the vocabulary. The total number of candidates becomes bv . For each of these candidates, the following score is calculated.

$$\log p(o_n) = \log p^{hyb}(o_n) + w_2 \log p^{lm}(o_n) \quad (6.2)$$

Here, $p(o_n)$ be the probability of output label o_n given the previous output labels, $p^{hyb}(o_n)$ be the score from hybrid CTC-attention system, $p^{lm}(o_n)$ be the language model score and w_2 be the language model weight. After calculating scores for each of the bv candidates, the top b candidates are considered for the next beam search step.

6.2.5 Contextual Rescoring

We use Labeled LDA for contextual relevance detection [81]. Regular LDA is an unsupervised algorithm that is not suitable for multi-label topic modeling. Unlike regular LDA, Labeled LDA allows the incorporation of a set of predefined topics. In our case, we consider each candidate sentence as a document and each contextual tag as a topic. We use label depth of 8 which is equal to the length of our contextual tags. We do not apply any dictionary pruning. Alpha and beta priors are set 0.1 and 0.01 respectively. Here, alpha represents document-topic density and beta represents topic-word density. We use a set of 37 contextual tags. We run training for 20 iterations.

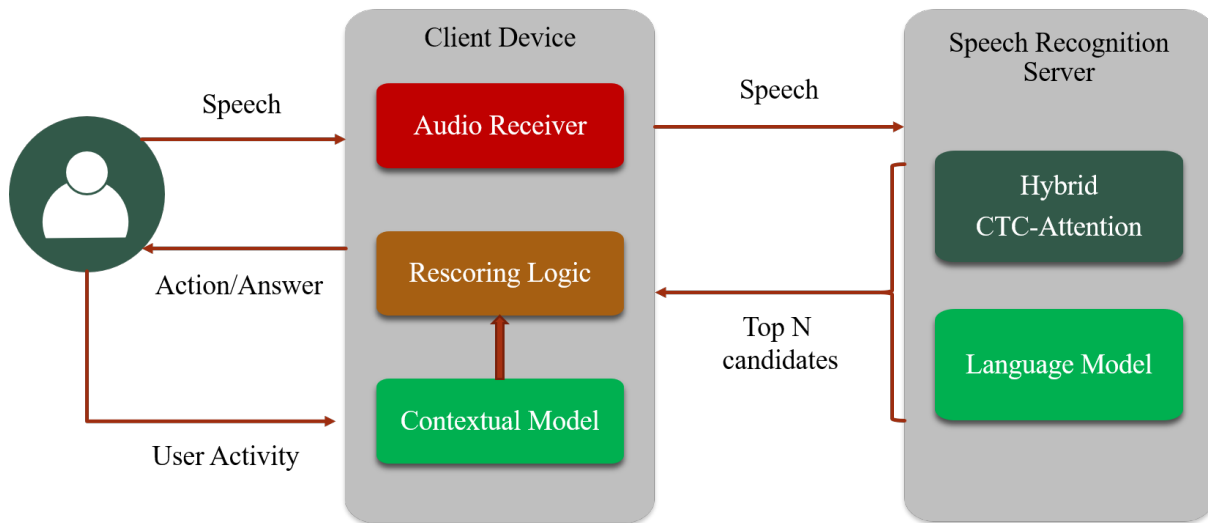


Figure 6.3: Contextual Rescoring in Client Device

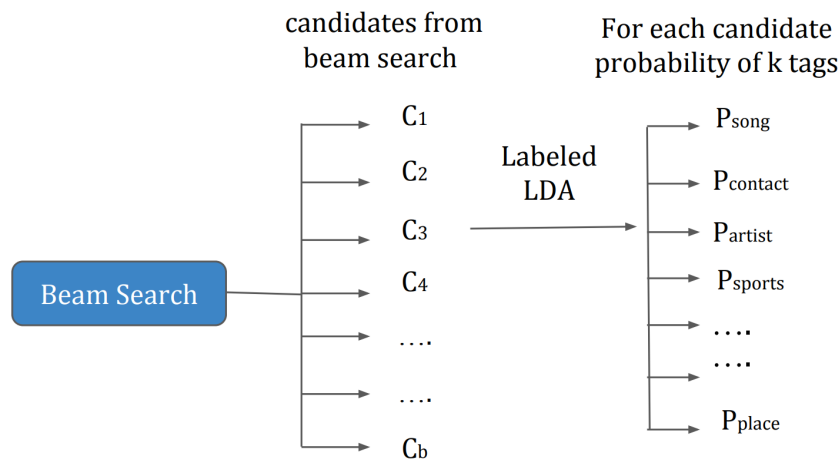


Figure 6.4: Rescoring System

Figure 6.3, 6.4 illustrates the context specific rescoring approach we used. After beam search, we have b candidates (assuming beam width of b) with their corresponding scores. First, we normalize the scores. Then, we apply topic modeling on each candidate. The output is a real-valued vector of length 37 (i.e. the number of context tags). Each value represents the relevance of the sentence to that particular context. If the detected context matches with any of the on-device contexts, a bias is added to this candidate's score. Added bias is proportional to contextual relevance. We use a context weight w_3 to tune the context-sensitivity of the system.

Algorithm 6 Contextual Rescoring

```

1:  $e2e \leftarrow$  Scores from End-to-End system
2:  $rv \leftarrow$  Contextual relevance vector
3:  $cl \leftarrow$  Current device contexts
4:  $w_3 \leftarrow$  Weight of contextual bias
5:  $\text{normalize}(e2e)$ 
6: for each  $c \in \text{candidateList}$  do
7:    $rv = \text{LabeledLDA}(c)$ 
8:   for each  $r \in rv$  do
9:     if  $r > \text{threshold}$  and  $r \in cl$  then
10:       $e2e[c] = e2e[c] + r \times w_3$ 
11:  $\text{output} \leftarrow$  Candidate with maximum  $e2e$  score

```

6.3 Contextual Corpus Management

6.3.1 Contextual Corpus Generation

We prepare a list of voice command templates from our domain study. These command templates contain entity tags. An example of a command template is ‘<contact> কে কল করো’ (Call <contact>). Here, <contact> is an entity tag. We have a list of 1700 voice command templates containing around 20 entity tags. The entity tags include contact names, app names, number, time & date, song, artist, writer, book, place, movie, actor, food, gadget, team, player, company, etc.

Figure 6.5 shows the contextual corpus generation process in user device. Whenever the user adds a new contact, all the command templates containing the entity tag <contact> are populated with the new contact name, and all the new sentences are added to the contextual text corpus. Similarly, when the user downloads a new song, all the command templates containing the entity tag <song> are populated and new sentences are added to the contextual text corpus. The contextual annotation of the newly formed sentences depends on the contextual annotation of the command template. Thus the contextual corpus gradually grows depending on the activity of the user. The labeled LDA system is periodically trained on the updated corpus. The personalized corpus of a particular user is fairly small, containing a few thousand sentences. It is possible to train the labeled LDA on a mid-range smartphone within a minute.

6.3.2 On-device Model Training

The information used for contextual model training is user’s private information such as contact list, app list, list of songs, previous search history, location data, etc. So we need to perform the contextual model training on client device to protect privacy (illustrated in figure 6.6). Therefore, we consider the complexity of training the labeled LDA-based system on the client device.

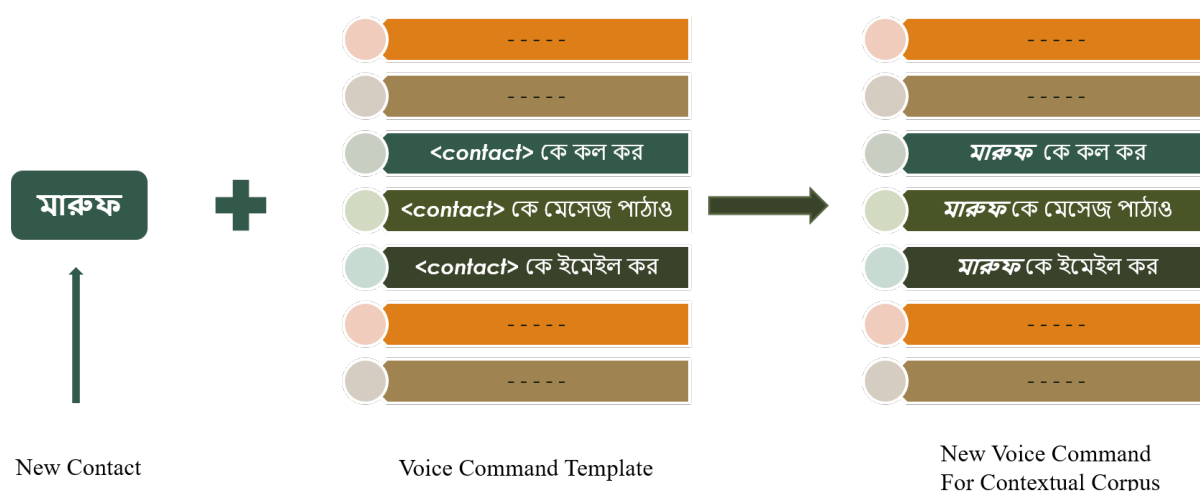


Figure 6.5: Contextual Corpus Extension

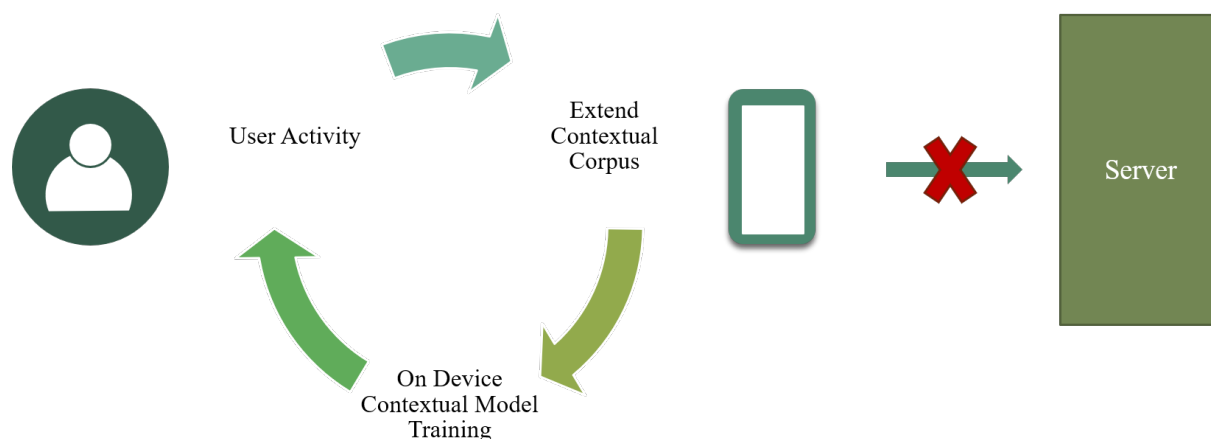


Figure 6.6: On Device Model Training

The main resource requirement of the labeled LDA training is the memory requirement. To be specific, the memory is mainly required to keep track of the topic distribution over document and word distribution over topic. So the memory requirement is dependent on the number of topics, the number of keywords, and the number of documents in the training set. Currently, the number of topics (i.e. contextual tags) is 37 and the number of relevant voice command keywords is around 2000. We do not expect these two parameters to grow larger because we already performed an elaborate domain study for voice commands. The memory requirement can only grow with the number of documents (i.e. context annotated voice commands) in the client device. For a typical user, we estimated the number of contacts, app lists, media list, etc. to estimate the size of the contextual corpus. We consider how many of our voice commands are associated with a particular entity tag, the number of possible values for each entity-tag for estimating the size of the contextual corpus.

For example, let us assume there are 500 people in the user’s contact list and there are 20 voice command templates with the entity tag <contact>. That means the contextual corpus will have 500×20 or 1000 voice commands related to this entity tag. We perform similar calculations for each possible entity tag. According to our estimation, the contextual corpus size can vary between 40,000 to 60,000 sentences for a typical user. So contextual model training will take a few megabytes of smartphone memory. This is not a big overhead because nowadays even mid-range smartphones have around 4 to 8 GB of RAM. Moreover, we can easily perform the model training at a time of the day when the user is usually inactive. If the size of the contextual corpus grows beyond our estimate, then we can easily reduce the size based on recency and frequency. For example, if a user downloads 10,000 songs in a device. We do not need to keep all the songs in our contextual corpus. We will only keep the songs user has listened to recently or frequently.

6.4 Dataset

6.4.1 Text Corpus

The text corpus was prepared after extensive crawling from various popular Bangla websites. We crawl from around 42 websites and collect 10 million sentences. After collection of raw sentences, we use text cleaning to remove non-Bangla sentences, punctuation, alphanumeric characters, inconsistency, duplicates from the collected text. Later, we normalize these sentences. We convert numbers to text, handle abbreviations, manage special numeric expressions in Bangla, normalize decimal point & percentage symbol, consider contact numbers, date, etc. We also add 1700 voice command specific texts collected from our domain study.

6.4.2 Speech Corpus

We consider all publicly available Bangla speech corpus as well as prepared a speech corpus of our own. The largest publicly available speech corpus is provided by Google [38]. It contains 217902 utterances from 505 speakers. Among the speakers, 323 of them are men and 182 women. The size of the speech corpus is approximately 220 hours. We also use our own open domain Bangla speech corpus. This corpus has around 150,000 utterances from 5190 speakers. There are 2680 male speakers and 2510 female speakers. The size of the speech corpus is approximately 510 hours. Overall, our open domain Bangla speech corpus has around 730 hours of speech data.

Bangla voice commands contain a set of technical words that are missing from all publicly available speech corpus. Also, the sentence structure of the voice commands is sometimes different from regular Bangla sentences. So we develop a speech corpus solely containing

Bangla voice commands. We prepare an Android app for speech data collection following the approach by [39]. We are able to collect 28973 sentences from 56 speakers using this application. Among the speakers, 34 are men and 22 are women. The size of this corpus is around 50 hours.

6.5 Experiments

6.5.1 Training Details

First, we train the hybrid CTC-attention based End-to-End system with our 780 hour speech corpus described above. The RNN based language model was trained with our Bangla text corpus containing 10 million sentences. The training of the End-to-End system takes around 72 hours and training of the RNN based language model takes around 18 hours. All experiments are done on a desktop with core i7 CPU, 16 GB RAM, Nvidia RTX 2070 GPU.

6.5.2 Test Set

Our test set contains 2000 voice command utterances with 7 speakers. We manually annotated the context for these utterances to prepare a context annotated test set. We refer to it as positive test set. We also randomly annotate context of these utterances to prepare a negative test set. the purpose of the negative test set is to test whether the system’s performance is affected with inaccurate context information.

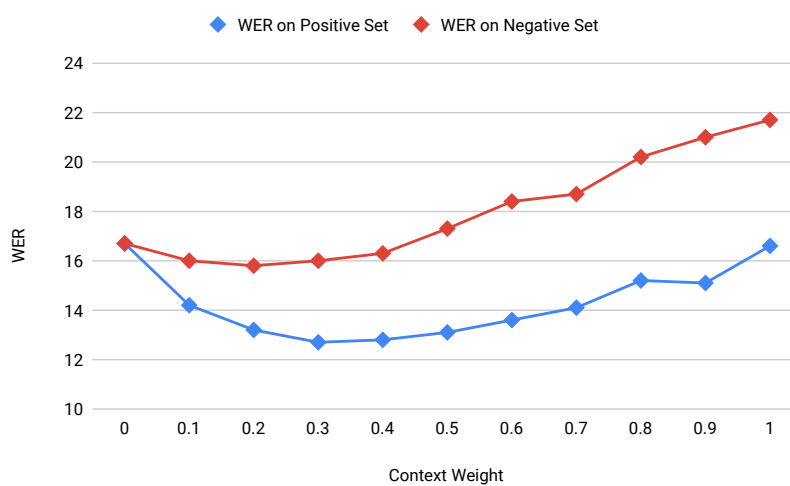
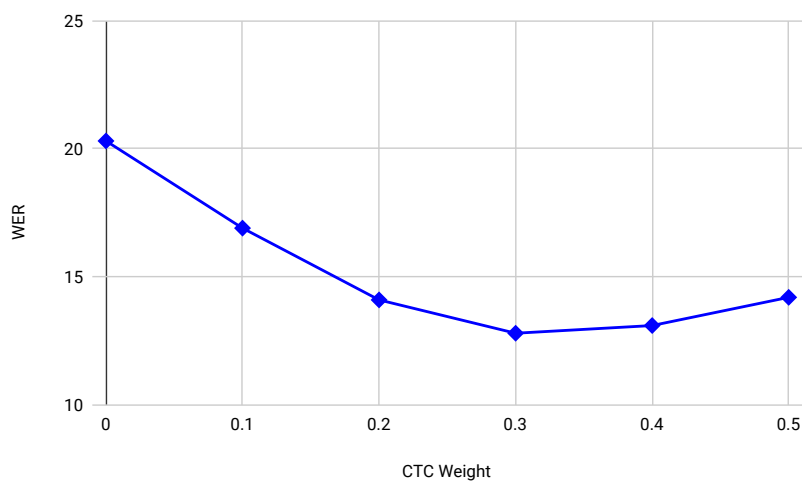
6.5.3 Results

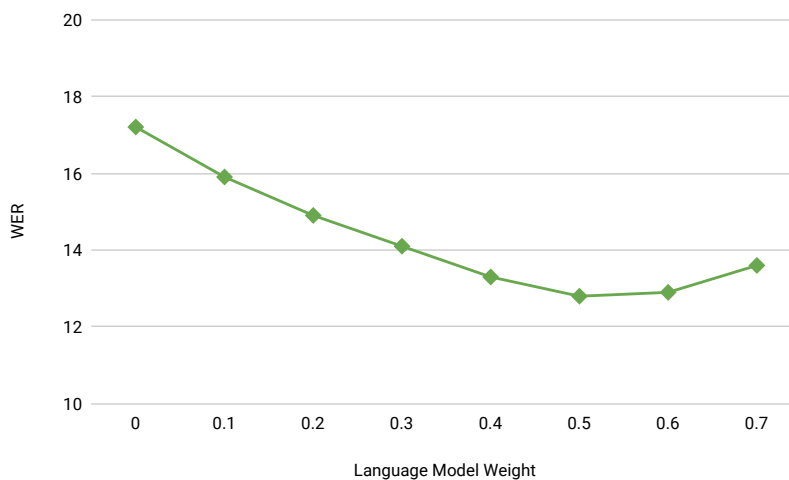
Table 6.1 shows Phoneme Error Rate (PER), Word Error Rate (WER) and Sentence Error Rate (SER) of our system in different setup. In our test set, the system using the character level RNN language model performs significantly better than the system using the word level RNN language model. The difference in performance is especially significant when test utterances contain out-of-vocabulary words. 700 out of 2000 test utterances contain one or more out-of-vocabulary words. For these utterances, WER was 26.6% and 46% for char-RNN and word-RNN respectively. We tried larger word-RNN networks such as 2 layer, 1000 LSTM cells and 2 layer, 2000 LSTM cells. Overall WER were 27.2% and 26.9% respectively. Our rescoring method outperforms trigram based contextual rescoring method.

The performance of our system for different voice command categories can be found in table 6.2. In particular, regular system commands are recognized with very high accuracy (WER 7.6%) because they contain no out-of-vocabulary words. Highest WER (19.2%) is found in the case of random queries because they often contain out-of-vocabulary and out-of-domain input.

Language Model	Rescoring	PER (%)	WER (%)	SER (%)
Word-RNN	None	6.9	27.9	44.9
	Trigram	6.4	25.7	42.4
	LLDA	6.0	23.8	40.3
Char-RNN	None	4.5	16.7	28.4
	Trigram	3.9	14.1	25.3
	LLDA	3.7	12.8	22.8

Table 6.1: Performance comparison

Figure 6.7: Effect of Context Weight w_3 Figure 6.8: Effect of CTC Weight w_1

Figure 6.9: Effect of Language Model Weight w_2

Category	WER
Regular System Commands	7.6
Numbers	9.2
Contacts	12.4
Place	13.5
Date and Time	15.7
Media	12.7
Random Queries	19.2

Table 6.2: WER for different Categories of Voice Command

Figure 6.7 shows the WER of the system for different values of context weight w_3 (algorithm 6). For the positive set, WER decreases upto $w_3 = 0.3$ then it starts to increase again. For the negative set, WER remains largely unaffected upto $w_3 = 0.3$ then it starts to increase almost linearly. For the experiment shown in Figure 6.7, we use CTC weight 0.3 and char-RNN language model with weight 0.5. Figure 6.8 and 6.9 shows the hyper-parameter tuning on validation set for CTC weight w_1 (equation 6.1) and language model weight w_2 (equation 6.2) respectively. We found best results using CTC weight 0.3 and language model weight 0.5.

Chapter 7

Semi-supervised Speech Recognition

An annotated speech corpus is an essential component for the development of an automatic speech recognition system (ASR). Speech corpus is a collection of audio files with corresponding text transcriptions. Manually developing a speech corpus of required size is a time consuming and monotonous task. It also requires some prerequisites like a recording environment, clear utterance, and additional information such as gender of speakers, etc. For achieving a large vocabulary continuous speech recognition we need approximately several hundred to few thousands of hours of speech corpus. Semi-supervised training can be a useful solution to tackle the hurdles related to speech corpus development. Semi-supervised training can provide us a way to exploit a huge collection of publicly available text as well as audio resources to improve the performance of an ASR.

In this work, we focus on improving an end-to-end speech recognition system for Bangladeshi Bangla using semi-supervised training. There are very few publicly available large speech corpora for Bangladeshi Bangla. Google released 229 hours of speech corpus for Bangladeshi Bangla [38]. But there are huge amounts of publicly available news audio files, audiobooks, recordings in Youtube and other media sources. There are a lot of text sources too like news websites, blogs, e-books, etc. Considering the abundance of unpaired audio and text data for Bangla language, a semi-supervised training method that can exploit both unpaired audio and text is very useful. Proper use of the unpaired data along with existing paired speech corpus can boost the performance of the Bangla ASR system.

Different researchers have tried different ways of incorporating this unlabelled, unannotated data for speech recognition. Our approach is similar to the approach used by [63]. We utilize an intermediate representation of speech and text data using a shared encoder network for semi-supervised training of the ASR system. Our contributions in this work are as follows:

- We propose a novel inter-domain loss function based on global encoding distance (GED loss) of speech and text data.

- Our proposed Global Encoding Distance (GED) loss for inter-domain features performs better than both the Gaussian KL-divergence loss proposed in [63] and Maximum Mean Discrepancy (MMD) loss proposed in [64]. Our loss function is more meaningful and intuitive in the context of unpaired audio and text data. The performance of the GED loss is more robust to minibatch size compared to Gaussian KL-divergence and MMD loss.
- To our best knowledge, this is the first work on Bangla language that incorporates semi-supervised training into deep learning based end-to-end ASR architecture.
- Using our semi-supervised training, we are able to exploit 1000 hours of unpaired audio data and 800K unpaired Bangla sentences. Our experiments show that our semi-supervised training with GED loss achieves WER of 31.9%, outperforming both the baseline end-to-end system with an external language model and semi-supervised method with MMD loss.

7.1 Related Works

Researchers have explored different methods of semi-supervised training for speech recognition. [51] investigate large-scale semi-supervised training to improve acoustic models for automatic speech recognition. They provide an empirical analysis of semi-supervised training with respect to transcription quality, data quality, filtering, etc. [52] pre-train the encoder-decoder network with unpaired speech and text. They use a large amount of unpaired audio to pre-train the encoder and synthesized audio from the unpaired text to pre-train the decoder. [53], [54] integrate active learning jointly with semi-supervised training in speech recognition system. [55] use transcribed multilingual data and semi-supervised training to circumvent the lack of sufficient training data for acoustic modeling. They train deep neural networks as data-driven feature front ends.

[56] use utterance-level and frame-level confidences for data selection during self-training. They find it beneficial to reduce the disproportion in amounts of paired and unpaired data by including the paired data several times in semi-supervised training. [57] describe the combination of deep neural networks and graph-based semi-supervised learning for acoustic modeling in speech recognition. [58] use a sparse auto-encoder to take advantage of both unlabelled and labeled data simultaneously through mini-batch stochastic gradient descent.

[59] try to improve the performance of a code-switching speech recognition system for Mandarin-English using semi-supervised training. They apply semi-supervised learning for lexicon learning as well as acoustic modeling. Similarly, [60] & [61] use untranscribed data for Luxembourgish & Lithuanian ASR respectively. [62] use a two-step training method to generalize the air traffic control speech recognizer. First, a baseline speech recognition system is trained using a paired speech corpus and it is used to transcribe publicly available unlabeled

data. The transcribed data is then filtered based on confidence scores and is used to retrain the acoustic model.

Recently, semi-supervised training has been proposed in the context of end-to-end ASR. [63] propose a shared encoder architecture for speech and text inputs that can encode both data from their respective domain to a common intermediate domain. They combine speech-to-text and text-to-text mapping by using the shared network to improve speech-to-text mapping. They propose an inter-domain loss function based on Gaussian KL-divergence which represents the dissimilarity between the encoded features of speech and text data. They later proposed an inter-domain loss function based on Maximum Mean Discrepancy [64]. In both cases, they assume that the encoded speech features in the current minibatch are sampled from one distribution and encoded text features in the current minibatch are sampled from a second distribution. The inter-domain loss is calculated based on the discrepancy of these two distributions. This approach has some weaknesses. The performance of this system varies based on the chosen minibatch size. Moreover, this approach does not take into account the variance of the current encoded features in the global context. We solve both problems by introducing a new inter-domain loss function based on global encoding distance.

7.2 Our System

In this section, we describe our baseline end-to-end architecture as well as semi-supervised architecture.

7.2.1 Baseline System

Our baseline system is an end-to-end ASR system based on the work of [76]. The architecture is shown in Figure 7.1. CTC and attention networks are combined in this architecture. Both networks share an encoder network. The shared encoder network has 6 layers of Bi-directional Long Short Term Memory (BLSTM) units. Each layer has 320 BLSTM units. A linear projection layer is connected to each BLSTM layer. The linear projection layer consists of 320 units. The decoder has 1 layer of unidirectional LSTM units. The number of LSTM units in this layer is 300. The scores from the attention network and the CTC network are combined during decoding. Let $p(c_t)$ be the probability of output label c_t at position t , given previous output labels and let w be the CTC weight.

$$\log p(c_t) = w \log p^{ctc}(c_t) + (1 - w) \log p^{att}(c_t) \quad (7.1)$$

As for the audio feature, we use 40 Mel-frequency cepstral coefficients (MFCC) per audio frame. We also consider their first and second-order temporal derivatives. So, we have 120 speech

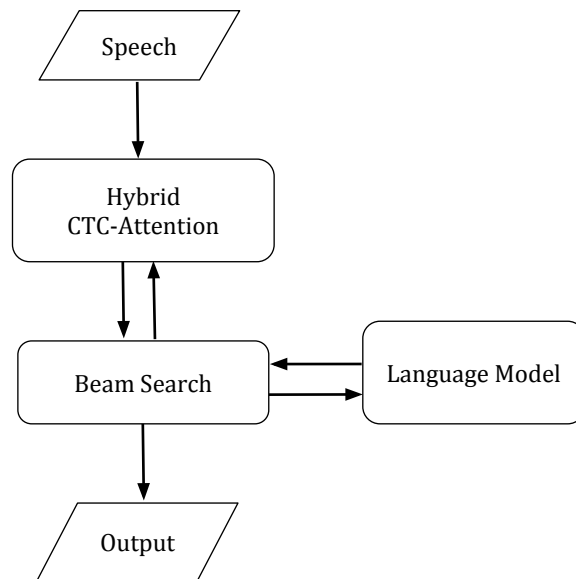


Figure 7.1: Baseline System

features per audio frame. These features are fed to the shared encoder and the attention decoder generates the character sequence.

We use a Recurrent Neural Network (RNN), based language model, in shallow fusion [80] with the baseline end-to-end architecture. We use both character level and word level RNN in our experiments. The character level RNN has 2 layers of LSTM, with each layer having 650 LSTM units. The word-level RNN has 1 hidden layer and this layer has 1000 LSTM units. For the word level RNN, we use most frequently used 65000 Bangla words as our vocabulary set.

7.2.2 Semi-supervised System

Our semi-supervised end-to-end speech recognition system for Bangla is based on the work of [63]. The semi-supervised architecture is shown in Figure 7.2. We use a shared encoder that encodes speech and text input sequences into a common intermediate domain. Speech feature sequences and text character sequences are very different in length. Also, speech features are continuous-valued vectors while text characters are discrete. We use a pyramid BLSTM network that performs sub-sampling on the speech feature sequence. The sub-sampling process shortens the length of the speech feature sequence. We use an embedding layer that converts the text character ids to continuous domain vectors. Thus, the speech and the text inputs become compatible with each other and they are both passed through a shared encoder containing BLSTM units.

Our encoder network has 6 layers of BLSTM cells. The size of each layer is 320 units. The decoder network has 1 layer of LSTM cells. The size of this layer is 300 units. First, this architecture is trained in a supervised manner using the paired speech corpus. Then, we perform

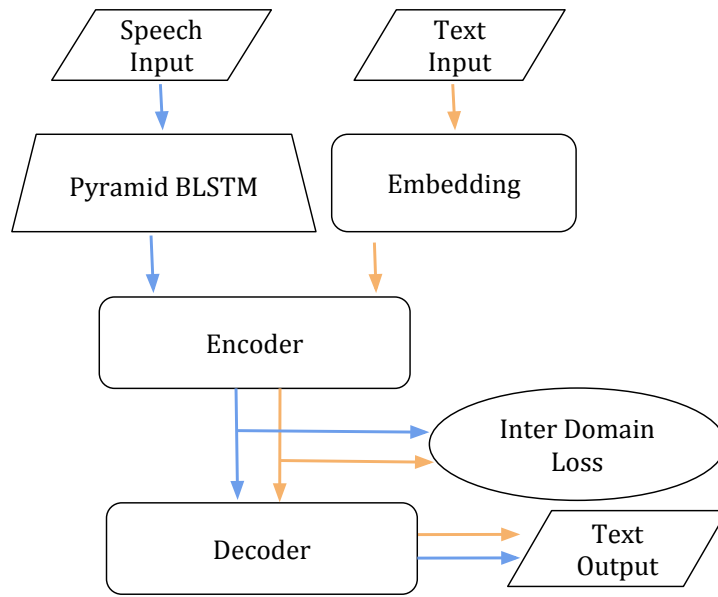


Figure 7.2: Semi-Supervised System

retraining using both paired and unpaired corpus. We use 3 different loss to guide semi-supervised retraining. They are the following:

Speech-to-text loss This is a conventional speech-to-text loss during supervised learning, which consists of a negative log-likelihood of the ground-truth text given by the encoded speech features. This loss is the combination of CTC and attention loss similar to the baseline system. We denote this loss as L_{sup} . The calculation of speech-to-text loss is shown in Equation 7.2. We use CTC weight w_1 to control the relative importance of CTC and attention loss.

Text-to-text auto-encoder loss This is the negative log-likelihood that the encoder-decoder architecture can reconstruct the output text from an unpaired text corpus. We denote this loss as L_{ae}

Inter-domain loss This is the dissimilarity between distributions of the encoded speech features and the encoded text features. We use global encoding distance as a measurement for our inter-domain loss. We denote this loss as L_{id} . More on this is described in section 7.3.

$$L_{sup} = w_1 L_{ctc} + (1 - w_1) L_{att} \quad (7.2)$$

$$L_{uns} = w_2 L_{id} + (1 - w_2) L_{ae} \quad (7.3)$$

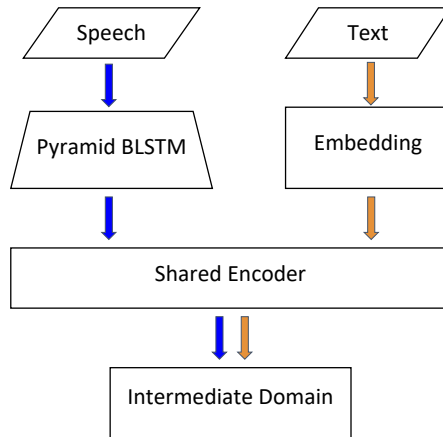


Figure 7.3: Overview of Encoding

$$L_{tot} = w_3 L_{sup} + (1 - w_3) L_{uns} \quad (7.4)$$

Equation 7.3 shows how the text auto-encoder loss and the inter-domain loss are combined to generate the unsupervised loss. We use speech text ratio parameter w_2 to control the relative importance of the text auto-encoder loss and the inter-domain loss. Then both the supervised loss L_{sup} and the unsupervised loss L_{uns} are combined to calculate the total loss L_{tot} (shown in Equation 7.4). Here, w_3 is the supervised loss ratio which controls the relative importance between the supervised and the unsupervised loss.

7.3 The Inter-Domain Loss

In this section, we describe our proposed inter-domain loss function.

7.3.1 Encoding Procedure

First, we pre-process the speech and text data in a way that they become compatible with each other. We reduce the length of the speech data by performing sub-sampling with a pyramid BLSTM unit. We also transform the text sequences into a continuous domain vector with an embedding layer. The pre-processed speech and text data are then absorbed by an encoder unit. The output of the encoder unit is considered as the inter-domain representation of the speech and text data. The overview of the encoding process is shown in Figure 7.3. Figure 7.4 shows the visualization of the encoded data using t-distributed stochastic neighbor embedding (t-SNE) [82].

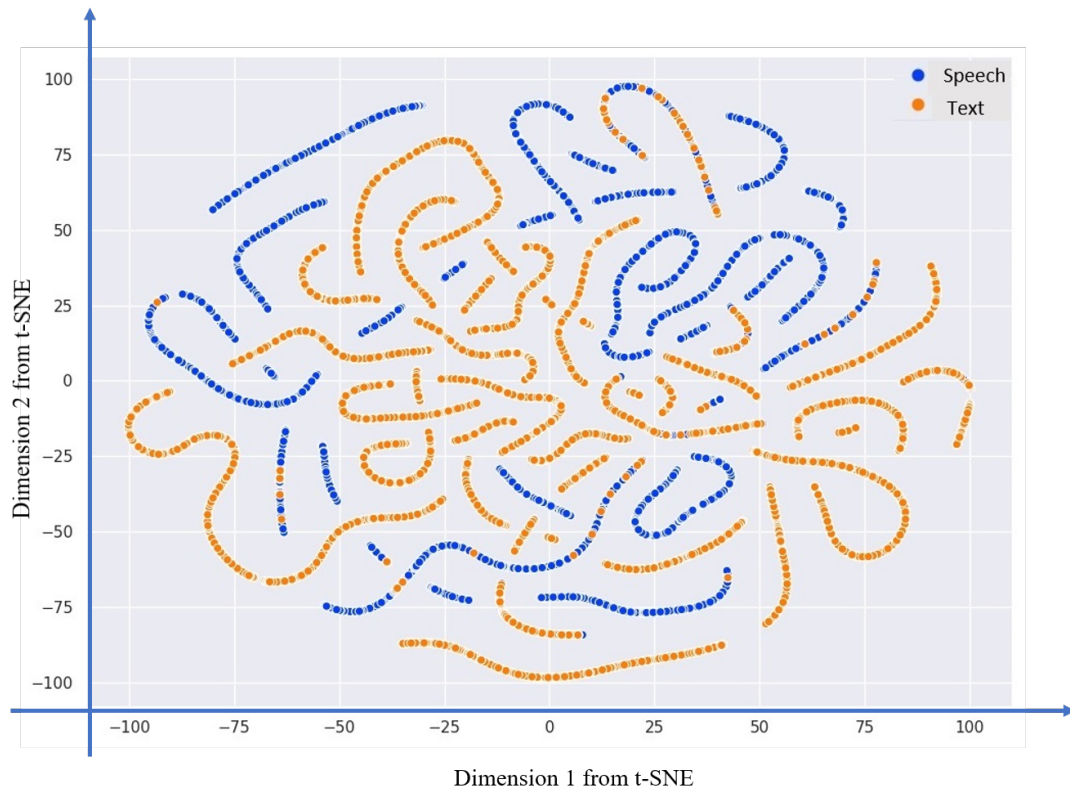


Figure 7.4: t-SNE Visualization of Encoded Data

7.3.2 Maximum Mean Discrepancy Loss

Here, we describe the MMD loss proposed by [64] and some of its limitations. A minibatch is formed by sampling the encoded features from unpaired speech and text data. All encoded speech features in this minibatch are considered to be from one underlying distribution. Similarly, all encoded text features from this minibatch are considered to be from another underlying distribution. Then Maximum Mean Discrepancy between these two distributions is calculated. A similar MMD calculation is repeated for the paired minibatch. Then the inter-domain loss is calculated by combining the MMD loss from the paired and the unpaired set, as shown in Algorithm 7.

This approach has some limitations because the distribution assumption is made only considering the unpaired data in the current minibatch. This loss calculation lacks the knowledge about global distribution, density, and variance of the unpaired data. Also, assuming a distribution based on the current minibatch makes the system unstable with respect to changing batch size. In other words, the system is not guaranteed to converge to the optimal solution for all minibatch sizes.

Algorithm 7 Computation of the MMD loss

```

1:  $N \leftarrow$  Number of samples
2:  $D \leftarrow$  dimension of encoded vector
3:  $H_{SP} \leftarrow$  encoded speech, paired minibatch
4:  $H_{TP} \leftarrow$  encoded text, paired minibatch
5:  $H_{SU} \leftarrow$  encoded speech, unpaired minibatch
6:  $H_{TU} \leftarrow$  encoded text, unpaired minibatch
7:  $H_{SP} \in \mathbb{R}^{N_{sp} \times D}$ ,  $H_{TP} \in \mathbb{R}^{N_{tp} \times D}$ 
8:  $H_{SU} \in \mathbb{R}^{N_{su} \times D}$ ,  $H_{TU} \in \mathbb{R}^{N_{tu} \times D}$ 
9: function LOSS( $H_{SP}, H_{TP}, H_{SU}, H_{TU}$ )
10:    $lp = \text{MMD}(H_{SP}, H_{TP})$ 
11:    $lu = \text{MMD}(H_{SU}, H_{TU})$ 
12:   return  $lp + lu$ 
13: function MMD( $H_S, H_T$ )
14:    $m_s = \sum_{i=1}^{N_s} \sum_{j=1}^{N_s} \sum_{k=1}^D H_{i,k}^S H_{j,k}^S$ 
15:    $m_t = \sum_{i=1}^{N_t} \sum_{j=1}^{N_t} \sum_{k=1}^D H_{i,k}^T H_{j,k}^T$ 
16:    $k_s = \frac{\sum_{i=1}^{N_s} \sum_{j=1}^{N_s} \exp(\sum_{k=1}^D H_{i,k}^S H_{j,k}^S - m_s)}{N_s^2}$ 
17:    $k_t = \frac{\sum_{i=1}^{N_t} \sum_{j=1}^{N_t} \exp(\sum_{k=1}^D H_{i,k}^T H_{j,k}^T - m_t)}{N_t^2}$ 
18:    $k_{s,t} = \frac{\sum_{i=1}^{N_s} \sum_{j=1}^{N_t} \exp(\sum_{k=1}^D H_{i,k}^S H_{j,k}^T - \frac{m_s}{2} - \frac{m_t}{2})}{N_s N_t}$ 
19:   return  $k_s + k_t - 2k_{s,t}$ 

```

7.3.3 Global Encoding Distance (GED) Loss

We have found that a significant performance gain can be made by exploiting the global distribution and variance of the encoded unpaired data. We pre-calculate the encoding for our entire unpaired dataset in every epoch and generate a representative matrix X for our unpaired set. X is calculated as follows. A set of neighboring points are repeatedly sampled from the encoded unpaired data. A representative mean is calculated for these neighboring points. X is the concatenation of all such neighboring means. Here, $X \in \mathbb{R}^{N_x \times D}$ where N_x is the number of representative means and D is the dimension of an encoded feature. The representative mean is used to reduce the size of the matrix X . This matrix X now functions as a global representing matrix for the unpaired set. Now the global encoding distance for an encoded vector v^i with respect to X is defined as follows:

$$d_i = \text{GED}(v^i | X) = \min_{j=1}^{N_x} \|e^j - v^i\| \quad (7.5)$$

Here, e^i is the i_{th} row of the matrix X ($e^i \in \mathbb{R}^{1 \times D}$) and it represents the i_{th} representing mean of

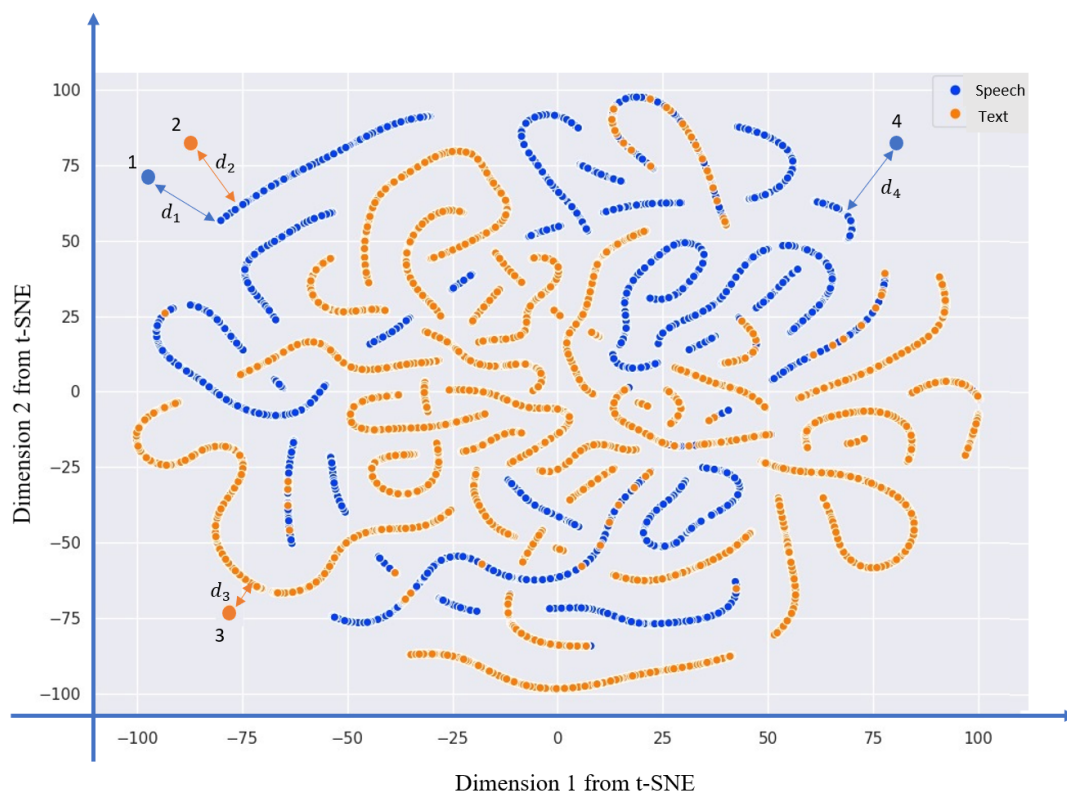


Figure 7.5: GED Loss

the unpaired set. The global encoding distance for four sample points is shown in Figure 7.5. For each point, the global encoding distance is the distance from this point to the closest representing mean in matrix X . The pseudocode for calculating inter-domain loss based on global encoding distance is shown in Algorithm 8.

Unlike MMD loss, our proposed loss function captures the dissimilarity between the encoded speech and text features with respect to the global representing matrix X . In addition to capturing the dissimilarity between the data in current minibatch, GED based loss also captures the variance of the encoded data in the global context. This system is less likely to suffer from any potential shortsightedness introduced by the assumption based on a few samples within a minibatch. Also, our system is more likely to converge to the optimal solution for any given minibatch size.

7.4 Corpus Description

In this section, we describe the corpus used for our experiments.

Algorithm 8 Computation of the GED loss

-
- 1: $N \leftarrow$ Number of samples
 - 2: $D \leftarrow$ dimension of encoded vector
 - 3: $H_{SP} \leftarrow$ encoded speech, paired minibatch
 - 4: $H_{TP} \leftarrow$ encoded text, paired minibatch
 - 5: $H_{SU} \leftarrow$ encoded speech, unpaired minibatch
 - 6: $H_{TU} \leftarrow$ encoded text, unpaired minibatch
 - 7: $H_{SP} \in \mathbb{R}^{N_{sp} \times D}, H_{TP} \in \mathbb{R}^{N_{tp} \times D}$
 - 8: $H_{SU} \in \mathbb{R}^{N_{su} \times D}, H_{TU} \in \mathbb{R}^{N_{tu} \times D}$
 - 9: **function** LOSS($H_{SP}, H_{TP}, H_{SU}, H_{TU}$)
 - 10: $l_{sp} = \sum_{i=1}^{N_{sp}} GED(H_{SP}^i | X)$
 - 11: $l_{tp} = \sum_{i=1}^{N_{tp}} GED(H_{TP}^i | X)$
 - 12: $l_{su} = \sum_{i=1}^{N_{su}} GED(H_{SU}^i | X)$
 - 13: $l_{tu} = \sum_{i=1}^{N_{tu}} GED(H_{TU}^i | X)$
 - 14: **return** $\frac{l_{sp} + l_{tp} + l_{su} + l_{tu}}{N_{sp} + N_{tp} + N_{su} + N_{tu}}$
-

7.4.1 Paired Speech Corpus

We use the corpus provided by [38] as our paired speech corpus. This corpus has around 229 hours of annotated speech data. The total number of utterances is around 217902 and the number of speakers is 505.

7.4.2 Unpaired Audio Data

The news recordings from a lot of Bangladeshi TV channels are available in the public domain. We mostly use these public domain news recordings as our audio source. After crawling the data, we split the audio files based on silence. We use 0.5 seconds as minimum silence duration and 0.0001 (between 0.0 and 1.0) as silence energy threshold. After silence based segmentation, we discard all audio files shorter than 3 seconds and longer than 9 seconds. Encoding audio files in the intermediate domain becomes easier when all audio files have a similar duration. After this, we have 1000 hours worth of unpaired audio corpus.

7.4.3 Unpaired Text Data

We use Bangla newspaper websites for preparing unpaired text corpus. We crawl around 40 Bangla websites. We use text cleaning on the collected data to remove non-Bangla symbols, punctuation, special characters, etc. We then perform text normalization. We convert all

numbers to their textual form, elaborate abbreviations, convert dates, etc. We apply the same text normalization on the text transcription of the paired dataset to maintain homogeneity among paired and unpaired corpus. After text cleaning and normalization, we discard all Bangla sentences that have fewer than 4 or greater than 10 words. Our text corpus has around 800K Bangla sentences.

7.5 Evaluations

In this section, we describe the experimental results.

7.5.1 Test Set

We separate 2000 utterances from the Google speech corpus as our test set. The test set has 5 speakers and covers various domains.

7.5.2 Training Details

At first, we train the CTC-attention network with the paired speech corpus. It takes around 10 hours in our setup. Then we retrain the model using the unpaired speech and text corpus along with the paired corpus. It takes around 20 hours. All experiments are performed on a hardware with a Core i7 processor, 16 GB Memory, NVIDIA GeForce GTX 1070 GPU. The important hyper-parameters of our system are shown in Table 7.1.

The training graph for the initial supervised training is shown in Figure 7.6. In this step, the system learns to minimize the CTC and the attention loss, effectively minimizing the supervised speech to text loss. The training graph for the retraining stage is shown in Figure 7.7. In this step, the system learns to minimize the text auto-encoder loss, as shown in Figure 7.7. The CTC and attention loss do not go through a big change in the retraining step because they have already been minimized. The inter-domain loss is calculated in an unsupervised manner, so the loss graph for the inter-domain loss does not fluctuate as much as the other loss terms throughout retraining.

7.5.3 Performance Comparison with External Language Model

To maintain fairness, we use the same unpaired text corpus to train the RNN language model in the baseline ASR model and the semi-supervised model. The only difference is, the semi-supervised model exploits the additional unpaired audio corpus. The RNN language model is used in shallow fusion with the baseline end-to-end system. Table 7.2 compares the Phoneme

Parameter	Value
Initialization	Uniform Distr
Encoder layers	6
Encoder layer size	320 (BLSTM)
Encoder projection layer size	320
Decoder layers	1
Decoder layer size	300 (LSTM)
Learning Rate	0.5
Batch size	24
CTC weight	0.3
Speech text ratio	0.1
Supervised loss ratio	0.9

Table 7.1: Hyper-parameter Description

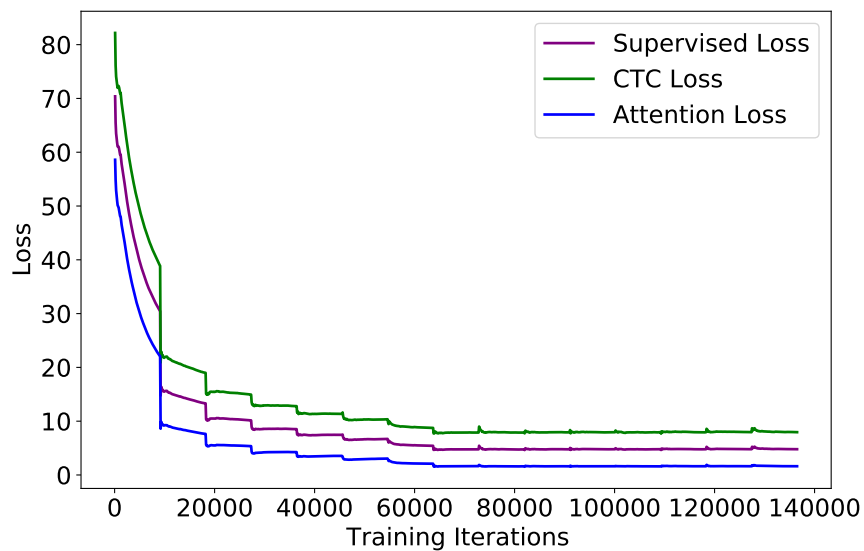


Figure 7.6: Supervised Training

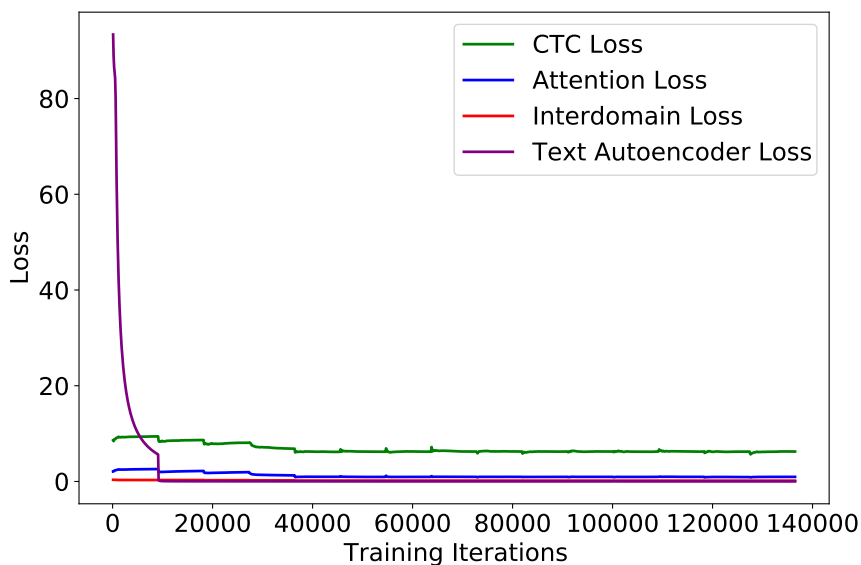


Figure 7.7: Semi-supervised Retraining

Model Type	Language Model	PER (%)	WER (%)	SER (%)
Baseline	None	12.6	37.0	64.6
	Word	12	33.8	60.2
	Char	11.4	32.5	58.5
Semi-supervised	None	11.3	31.9	58
	Word	11.6	30.5	55.5
	Char	11.4	28.9	52.6

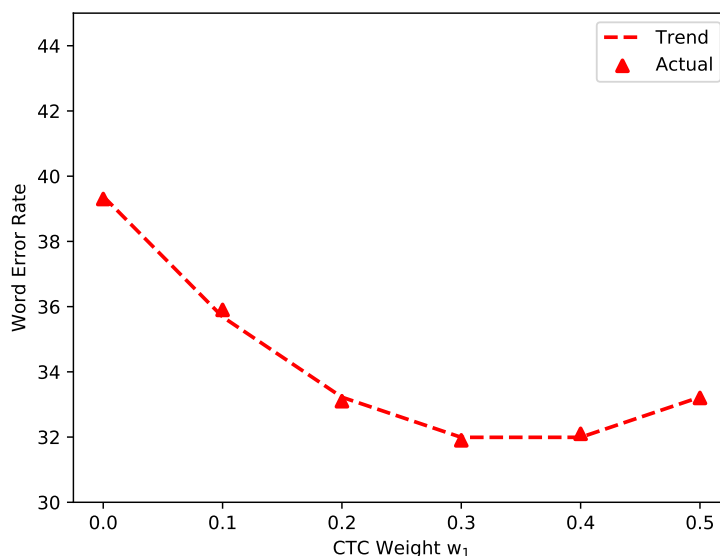
Table 7.2: Performance Comparison with Baseline

Error Rate (PER), Word Error Rate (WER), and Sentence Error Rate (SER) of our system with the baseline system with an external language model.

When we do not use any language model, the baseline end-to-end system achieves WER of 37%. Adding a word-level RNN language model improves the WER to 33.8%. The best accuracy in the baseline setup is achieved by the character level RNN where the WER is 32.5%. The character level RNN performs better than the word level RNN probably due to the presence of out-of-vocabulary words in the test set. The semi-supervised end-to-end system that exploits unpaired audio and text data outperforms all baseline setup and achieves WER of 31.9%. Using language model for decoding, the performance of the semi-supervised system improves further to 28.9%.

Inter-Domain Loss	PER (%)	WER (%)	SER (%)
Gaussian KL	11.9	34.0	60.8
MMD	11.4	32.7	59.1
GED	11.3	31.9	58

Table 7.3: Performance of Inter-Domain Loss

Figure 7.8: Effect of CTC Weight w_1

7.5.4 Performance Comparison of Inter-domain Loss

Table 7.3 shows the performance of the semi-supervised system for different inter-domain loss. Our proposed inter-domain loss based on global encoding distance achieves WER of 31.9% and SER of 58%, outperforming both Gaussian KL and MMD loss.

7.5.5 Effect of CTC Weight

Figure 7.8 shows the effect of the CTC weight w_1 (Equation 7.2) on the performance of our system. We found the best results when using CTC weight of 0.3. The tuning of the hyper-parameters is performed on a separate validation set.

7.5.6 Effect of Speech Text Ratio

Figure 7.9 shows the effect of the speech text ratio w_2 (Equation 7.3) on the performance of our system. We found the best results when using speech text ratio of 0.1.

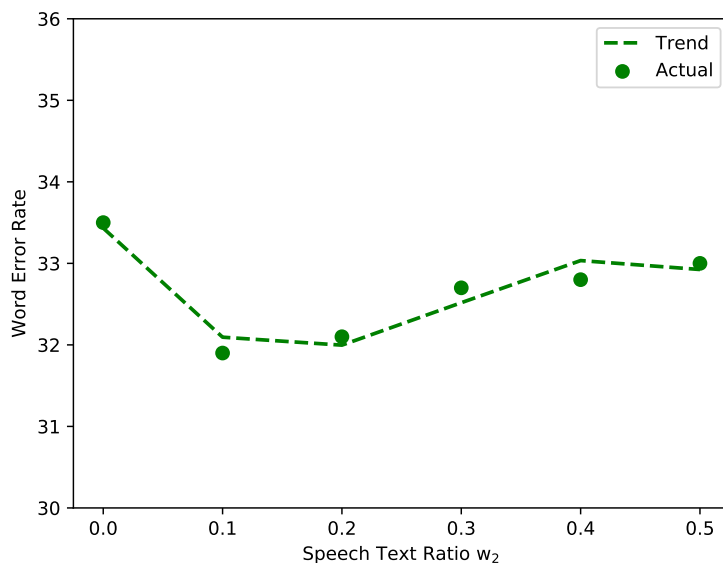


Figure 7.9: Effect of Speech Text Ratio w_2

7.5.7 Effect of Supervised Loss Ratio

Figure 7.10 shows the effect of the supervised loss ratio w_3 (Equation 7.4) on the performance of our system. We found the best results when using supervised loss ratio of 0.9.

7.5.8 Effect of Batch Size

Figure 7.11 shows the performance of the semi-supervised system with respect to batch size. The performance of the semi-supervised system with MMD loss decreases with smaller batch sizes. Our proposed GED loss is more robust to batch size and more likely to converge to the optimal solution even for small batch size.

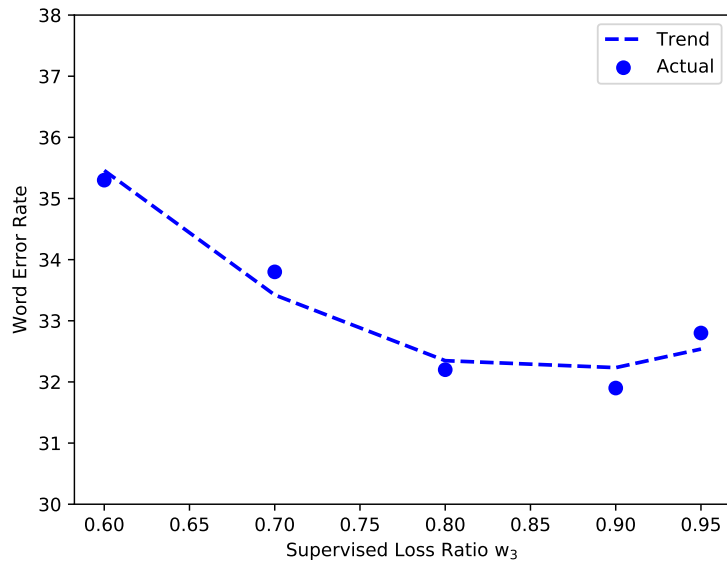
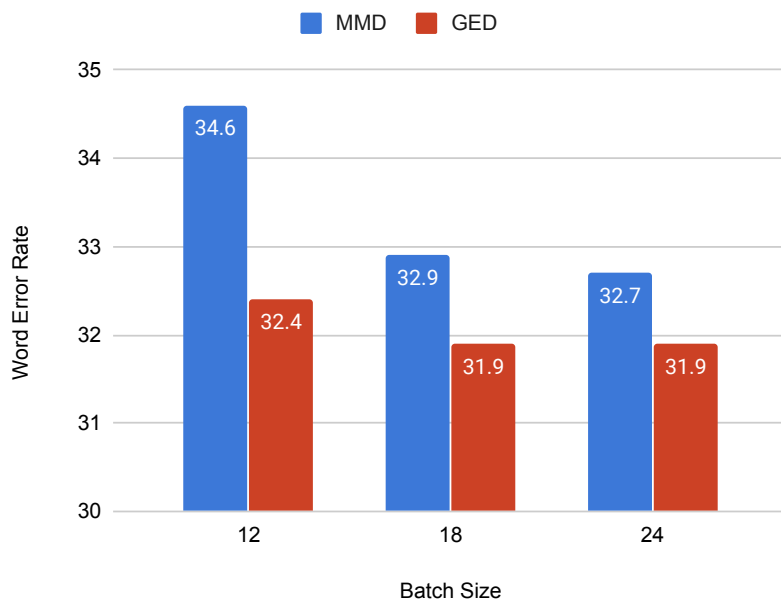
Figure 7.10: Effect of Supervised Loss Ratio w_3 

Figure 7.11: Effect of Batch Size

Chapter 8

Evaluation of ASR System

In this chapter, we evaluate our ASR system on open domain and voice command specific Bangla ASR tasks.

8.1 Experiments on Open Domain ASR Task

8.1.1 Dataset

Text Corpus

The text corpus was prepared after extensive crawling from various popular Bangla websites. We crawl from around 42 websites and collect 10 million sentences. After collection of raw sentences, we use text cleaning to remove non-Bangla sentences, punctuation, alphanumeric characters, inconsistency, duplicates from the collected text. Later, we normalize these sentences. We convert numbers to text, handle abbreviations, manage special numeric expressions in Bangla, normalize decimal point & percentage symbol, consider contact numbers, date, etc.

Speech Corpus

Table 8.1 shows the summary of our speech corpus for open domain ASR task. We use 220 hours of Google speech corpus, 510 hours of transcribed and verified speech corpus and 450 hours of synthesized speech corpus. The overall size of the corpus is around 1180 hours.

8.1.2 Training Details

First, we train the hybrid CTC-attention based End-to-End system with our speech corpus described above. The RNN based language model was trained with our Bangla text corpus

Aspect	Google	Transcribed	Synthesized	Total
Num of Utt	217902	150,000	1.64M	2M
Num of Speaker	505	5190	1	5696
Male Speaker	323	2680	N/A	3003
Female Speaker	182	2510	N/A	2692
Size (Hours)	220	510	450	1180

Table 8.1: Speech Corpus

containing 10 million sentences. The training of the End-to-End system takes around 120 hours and training of the RNN based language model takes around 18 hours. All experiments are done on a desktop with core i7 CPU, 16 GB RAM, Nvidia RTX 2070 GPU.

8.1.3 Test Set

Our test set contains 13000 open domain utterances separated from our speech corpus. We do not consider synthetic speech for test set. There are 35 speakers in the test set. Because the utterances of the test set are collected from publicly available sources, the signal to noise ratio (SNR) of the test set varies a lot. According to our estimation, the worst, best, mean and median of SNR from crowd-sourced speech data are -2.1, 57.8, 15.1, 12.6 respectively.

8.1.4 Results

We use two different models for evaluating each speech corpus. As our first model, we use traditional HMM-GMM based recipe from Kaldi [83]. HMM-GMM based system requires a lexicon that contains the phonetic transcriptions for all words in the vocabulary. The same lexicon was used for evaluating both speech corpora. We use our previously developed lexicon [84]. The lexicon contains 95000 transcribed Bangla words. An N-gram based language model is used with HMM-GMM based model.

As our second model, we use a deep learning-based end to end speech recognition system that uses a hybrid of CTC and Attention mechanism. We follow the approach of [76]. We use four BLSTM layers in the encoder network. The number of BLSTM cells in each layer is 320. Each BLSTM layer is connected to a linear projection layer with 320 units. The decoder network has 1 layer with 300 unidirectional LSTM units. We use a Recurrent Neural Network-based language model in shallow fusion with the CTC-Attention network. We use word level RNN. The RNN has 1 layers with 1000 LSTM units in each layer. We get the best performance when using CTC weight 0.3 with the language model weight 0.5.

Table 8.2 shows the performance of the ASR system for different combination of training datasets and models. Best performance is achieved when we use both transcribed and synthesized corpus along with Google’s dataset. For example, when using CTC-Attention

Train Set	Model	LM	WER (%)
Google	HMM-GMM	N-gram	30.95
	CTC-Attention	RNN	26.0
Google +Transcribed	HMM-GMM	N-gram	27.20
	CTC-Attention	RNN	23.0
Google +Synthesized	HMM-GMM	N-gram	31.40
	CTC-Attention	RNN	26.6
Google+ Transcribed+Synthesized	HMM-GMM	N-gram	24.38
	CTC-Attention	RNN	20.2

Table 8.2: Evaluation of ASR performance

Beam Width	Decoding Speed (WPM)	WER
8	108	24.3
12	77	21.2
16	72	20.2
20	54	19.5

Table 8.3: Decoding Speed

network, the combined corpus achieves WER of 20.2%. This system outperforms the same model trained on Google’s dataset only, which shows WER of 26.0%. When we use Google+Transcribed corpus the WER is 23.0%. This shows the effectiveness of our iterative corpus generation approach. When use Google+Synthesized corpus, the system actually performs even worse than the system trained on Google corpus alone. Possibly, it happens because the size of synthesized corpus is larger than Google corpus. The overwhelming size of the synthesized corpus leads to excessive bias. But presence of synthesized corpus improves the ASR performance when we use all three corpus combined.

Table 8.3 shows the decoding speed of the end-to-end ASR system (+RNNLM) with respect to a changing beam width. Beam width can be used to find appropriate compromise between decoding speed and ASR accuracy.

8.2 Experiments on Voice Command Task

8.2.1 Dataset

Text Corpus

We use the same text corpus mentioned in 8.1.1. We also add 1700 voice command specific texts collected from our domain study.

Aspect	Google	Voice Command	Transcribed	Total
Num of Utt	217902	28973	150,000	396875
Num of Speaker	505	56	5190	5751
Male Speaker	323	34	2680	3037
Female Speaker	182	22	2510	2714
Size (Hours)	220	50	510	780

Table 8.4: Speech Corpus

Speech Corpus

Table 8.4 shows the summary of our speech corpus for voice command recognition task. We use 220 hours of Google speech corpus, 510 hours of transcribed and verified speech corpus and 50 hours of voice command corpus. The overall size of the corpus is around 780 hours.

8.2.2 Training Details

First, we train the hybrid CTC-attention based End-to-End system with our speech corpus described above. The RNN based language model was trained with our Bangla text corpus containing 10 million sentences. The training of the End-to-End system takes around 72 hours and training of the RNN based language model takes around 18 hours. All experiments are done on a desktop with core i7 CPU, 16 GB RAM, Nvidia RTX 2070 GPU.

8.2.3 Test Set

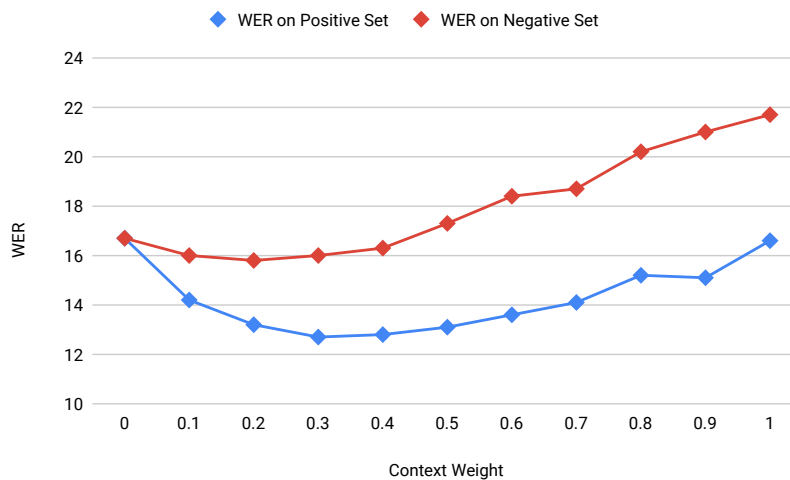
Our test set contains 2000 voice command utterances with 7 speakers. We manually annotated the context for these utterances to prepare a context annotated test set. We refer to it as positive test set. We also randomly annotate context of these utterances to prepare a negative test set. the purpose of the negative test set is to test whether the system’s performance is affected with inaccurate context information.

8.2.4 Results

Table 8.5 shows Phoneme Error Rate (PER), Word Error Rate (WER) and Sentence Error Rate (SER) of our system in different setup. In our test set, the system using the character level RNN language model performs significantly better than the system using the word level RNN language model. The difference in performance is especially significant when test utterances contain out-of-vocabulary words. 700 out of 2000 test utterances contain one or more out-of-vocabulary words. For these utterances, WER was 26.6% and 46% for char-RNN and word-RNN respectively. We tried larger word-RNN networks such as 2 layer, 1000 LSTM cells and

Language Model	Rescoring	PER (%)	WER (%)	SER (%)
Word-RNN	None	6.9	27.9	44.9
	Trigram	6.4	25.7	42.4
	LLDA	6.0	23.8	40.3
Char-RNN	None	4.5	16.7	28.4
	Trigram	3.9	14.1	25.3
	LLDA	3.7	12.8	22.8

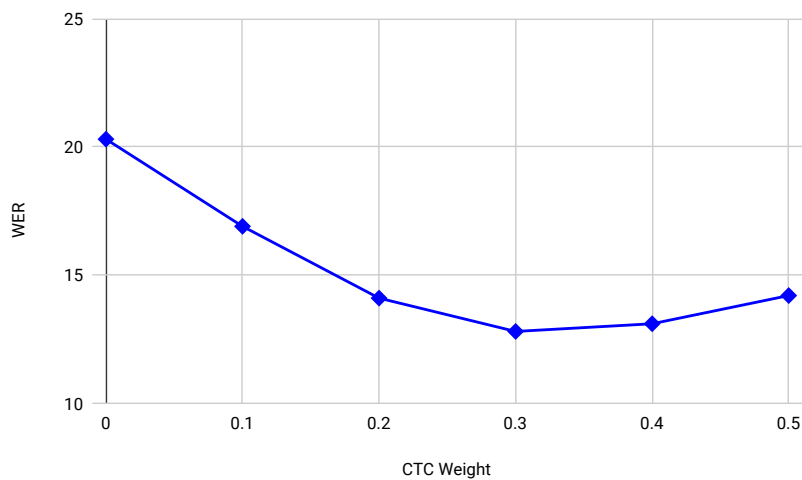
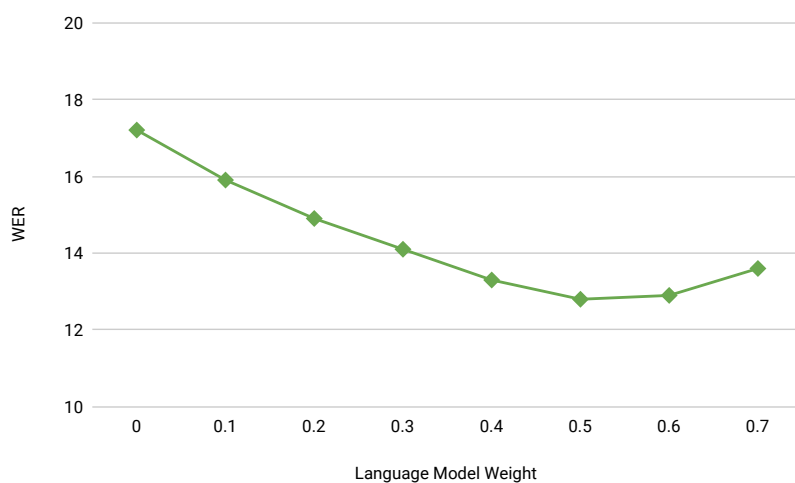
Table 8.5: Performance on Voice Command Task

Figure 8.1: Effect of Context Weight w_3

2 layer, 2000 LSTM cells. Overall WER were 27.2% and 26.9% respectively. Our rescoring method outperforms trigram based contextual rescoring method.

The performance of our system for different voice command categories can be found in table 8.6. In particular, regular system commands are recognized with very high accuracy (WER 7.6%) because they contain no out-of-vocabulary words. Highest WER (19.2%) is found in the case of random queries because they often contain out-of-vocabulary and out-of-domain input.

Figure 8.1 shows the WER of the system for different values of context weight w_3 (algorithm 6). For the positive set, WER decreases upto $w_3 = 0.3$ then it starts to increase again. For the negative set, WER remains largely unaffected upto $w_3 = 0.3$ then it starts to increase almost linearly. For the experiment shown in Figure 8.1, we use CTC weight 0.3 and char-RNN language model with weight 0.5. Figure 8.2 and 8.3 shows the hyper-parameter tuning on validation set for CTC weight w_1 (equation 6.1) and language model weight w_2 (equation 6.2) respectively. We found best results using CTC weight 0.3 and language model weight 0.5.

Figure 8.2: Effect of CTC Weight w_1 Figure 8.3: Effect of Language Model Weight w_2

Category	WER
Regular System Commands	7.6
Numbers	9.2
Contacts	12.4
Place	13.5
Date and Time	15.7
Media	12.7
Random Queries	19.2

Table 8.6: WER for different Categories of Voice Command

Chapter 9

Conclusion and Future Work

In this chapter, we provide a summary of our research work, related outcomes, and future research direction.

9.1 Summary of Research Work

We performed research on traditional as well as state-of-the-art End-to-end speech recognition architectures. We developed a web application and Android application related to the corpus buildup system. We developed a web crawler, text cleaner, text normalizer, word frequency analysis tool, Grapheme-to-Phoneme conversion tool for our ASR. We have developed an improved lexicon that has around 100K most frequently used Bangla words and considers critical cases for G2P conversion in Bangla language. We were able to generate a text corpus with 10 million Bangla sentences, We proposed an efficient method for automated speech corpus development and developed a speech corpus with 1010 hours of annotated speech. We solved the out-of-vocabulary problem of Bangla ASR system using synthetic speech data generation. We proposed a novel approach for context-specific optimization of voice commands which is based on multi-label topic modeling. We have proposed a novel semi-supervised ASR training method for exploiting unpaired audio data effectively.

9.2 Research Outcome

The outcomes of our study are as follows:

- An improved lexicon that has around 100K most frequently used Bangla words and considers critical cases for G2P conversion in Bangla language.
- An efficient data collection tool for speech corpus development.

- A Bangla Speech Corpus containing both domain-independent and domain-specific utterances. The overall size of the corpus is 1010 hours.
- A Bangla text corpus containing 10 million Bangla sentences.
- A context annotated voice command corpus. The corpus has 1700 voice command templates and includes all necessary voice commands related to smartphones, home appliances, automotive and office work accessories.
- A novel approach for context-specific optimization of voice commands which is based on multi-label topic modeling.
- A novel semi-supervised ASR training method for exploiting unpaired audio data effectively.

9.3 Future Research Direction

In the future, we need to find more effective ways of learning from unpaired speech data. More advanced semi-supervised or fully unsupervised ASR training approaches need to be explored. Another important research direction would be multi-modal speech recognition such as audio-visual ASR. It may be possible to improve the ASR performance significantly by exploiting the available visual cues such as speaker lip movement or body language.

References

- [1] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: an asr corpus based on public domain audio books,” in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206–5210, IEEE, 2015.
- [2] S. F. Chen, “Conditional and joint models for grapheme-to-phoneme conversion,” in *Eighth European Conference on Speech Communication and Technology*, 2003.
- [3] K. Yao and G. Zweig, “Sequence-to-sequence neural net models for grapheme-to-phoneme conversion,” *arXiv preprint arXiv:1506.00196*, 2015.
- [4] air. <https://svn.code.sf.net/p/cmuspinyin/code/branches/cmudict/cmudict-0.7b>, 2015.
- [5] Y. K. Thu, W. P. Pa, Y. Sagisaka, and N. Iwahashi, “Comparison of grapheme-to-phoneme conversion methods on a myanmar pronunciation dictionary,” in *Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSANLP2016)*, pp. 11–22, 2016.
- [6] M. Bisani and H. Ney, “Joint-sequence models for grapheme-to-phoneme conversion,” *Speech communication*, vol. 50, no. 5, pp. 434–451, 2008.
- [7] J. R. Novak. <https://github.com/AdolfVonKleist/Phonetisaurus>, 2012.
- [8] J. R. Novak, N. Minematsu, and K. Hirose, “Failure transitions for joint n-gram models and g2p conversion.,” in *INTERSPEECH*, pp. 1821–1825, 2013.
- [9] R. Caruana, “Multitask learning,” *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.
- [10] S. Jiampojamarn, G. Kondrak, and T. Sherif, “Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion,” in *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pp. 372–379, 2007.
- [11] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

- [12] K. Rao, F. Peng, H. Sak, and F. Beaufays, “Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks,” in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pp. 4225–4229, IEEE, 2015.
- [13] C. Schnober, S. Eger, E.-L. D. Dinh, and I. Gurevych, “Still not there? comparing traditional sequence-to-sequence models to encoder-decoder neural networks on monotone string translation tasks,” *arXiv preprint arXiv:1610.07796*, 2016.
- [14] Y. Tsvetkov, S. Sitaram, M. Faruqui, G. Lample, P. Littell, D. Mortensen, A. W. Black, L. Levin, and C. Dyer, “Polyglot neural language models: A case study in cross-lingual phonetic representation learning,” *arXiv preprint arXiv:1605.03832*, 2016.
- [15] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- [16] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, *et al.*, “Google’s neural machine translation system: Bridging the gap between human and machine translation,” *arXiv preprint arXiv:1609.08144*, 2016.
- [17] M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, *et al.*, “Google’s multilingual neural machine translation system: enabling zero-shot translation,” *arXiv preprint arXiv:1611.04558*, 2016.
- [18] S. Toshniwal and K. Livescu, “Jointly learning to align and convert graphemes to phonemes with neural attention models,” in *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pp. 76–82, IEEE, 2016.
- [19] 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*, pp. 5998–6008, 2017.
- [20] A. B. Mosaddeque, N. UzZaman, and M. Khan, “Rule based automated pronunciation generator,” 2006.
- [21] F. Alam, S. M. Habib, and M. Khan, “Bangla text to speech using festival,” in *Conference on Human Language Technology for Development*, pp. 154–161, 2011.
- [22] J. Basu, T. Basu, M. Mitra, and S. K. D. Mandal, “Grapheme to phoneme (g2p) conversion for bangla,” in *Speech Database and Assessments, 2009 Oriental COCOSDA International Conference on*, pp. 66–71, IEEE, 2009.

- [23] K. Ghosh, R. V. Reddy, N. Narendra, S. Maity, S. Koolagudi, and K. Rao, “Grapheme to phoneme conversion in bengali for festival based tts framework,” in *8th international conference on natural language processing (ICON)*, Macmillan Publishers, 2010.
- [24] A. Gutkin, L. Ha, M. Jansche, K. Pipatsrisawat, and R. Sproat, “Tts for low resource languages: A bangla synthesizer,” in *LREC*, 2016.
- [25] S. A. Chowdhury, F. Alam, N. Khan, and S. R. Noori, “Bangla grapheme to phoneme conversion using conditional random fields,” in *Computer and Information Technology (ICCIT), 2017 20th International Conference of*, pp. 1–6, IEEE, 2017.
- [26] F. Mana, P. Massimino, and A. Pacchiotti, “Using machine learning techniques for grapheme to phoneme transcription,” in *Seventh European Conference on Speech Communication and Technology*, 2001.
- [27] Y.-B. Kim and B. Snyder, “Universal grapheme-to-phoneme prediction over latin alphabets,” in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 332–343, Association for Computational Linguistics, 2012.
- [28] A. Deri and K. Knight, “Grapheme-to-phoneme models for (almost) any language,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, pp. 399–408, 2016.
- [29] B. Milde, C. Schmidt, and J. Köhler, “Multitask sequence-to-sequence models for grapheme-to-phoneme conversion,” *Proc. Interspeech 2017*, pp. 2536–2540, 2017.
- [30] P. J. Jang and A. G. Hauptmann, “Improving acoustic models with captioned multimedia speech,” in *Proceedings IEEE International Conference on Multimedia Computing and Systems*, vol. 2, pp. 767–771, IEEE, 1999.
- [31] E. Lakomkin, S. Magg, C. Weber, and S. Wermter, “Kt-speech-crawler: Automatic dataset construction for speech recognition from youtube videos,” *arXiv preprint arXiv:1903.00216*, 2019.
- [32] A. Mansikkaniemi, P. Smit, M. Kurimo, *et al.*, “Automatic construction of the finnish parliament speech corpus,” in *INTERSPEECH*, pp. 3762–3766, 2017.
- [33] I. R. Helgadóttir, R. Kjaran, A. B. Nikulásdóttir, and J. Guðhnason, “Building an asr corpus using althingi’s parliamentary speeches,” in *INTERSPEECH*, pp. 2163–2167, 2017.
- [34] T. Patel, D. Krishna, N. Fathima, N. Shah, C. Mahima, D. Kumar, and A. Iyengar, “An automatic speech transcription system for manipuri language,” *Show and Tell Session in INTERSPEECH, Hyderabad*, 2018.

- [35] M. M. H. Nahid, M. Islam, B. Purkaystha, M. S. Islam, *et al.*, “Comprehending real numbers: Development of bengali real number speech corpus,” *arXiv preprint arXiv:1803.10136*, 2018.
- [36] M. F. Khan and M. A. Sobhan, “Construction of large scale isolated word speech corpus in bangla,” *Global Journal of Computer Science and Technology*, 2018.
- [37] M. F. Khan and M. A. Sobhan, “Creation of connected word speech corpus for bangla speech recognition systems,” *Asian Journal of Research in Computer Science*, pp. 1–6, 2018.
- [38] O. Kjartansson, S. Sarin, K. Pipatsrisawat, M. Jansche, and L. Ha, “Crowd-sourced speech corpora for javanese, sundanese, sinhala, nepali, and bangladeshi bengali,” in *Proc. The 6th Intl. Workshop on Spoken Language Technologies for Under-Resourced Languages*, pp. 52–55, 2018.
- [39] T. Hughes, K. Nakajima, L. Ha, A. Vasu, P. J. Moreno, and M. LeBeau, “Building transcribed speech corpora quickly and cheaply for many languages,” in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.
- [40] P. Aleksic, M. Ghodsi, A. Michaely, C. Allauzen, K. Hall, B. Roark, D. Rybach, and P. Moreno, “Bringing contextual information to google speech recognition,” in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.
- [41] K. Hall, E. Cho, C. Allauzen, F. Beaufays, N. Coccaro, K. Nakajima, M. Riley, B. Roark, D. Rybach, and L. Zhang, “Composition-based on-the-fly rescoring for salient n-gram biasing.” <https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43816.pdf>, 2015.
- [42] L. Velikovich, I. Williams, J. Scheiner, P. Aleksic, P. Moreno, and M. Riley, “Semantic lattice processing in contextual automatic speech recognition for google assistant,” *Proc. Interspeech 2018*, pp. 2222–2226, 2018.
- [43] A. H. Michaely, M. Ghodsi, Z. Wu, J. Scheiner, and P. Aleksic, “Unsupervised context learning for speech recognition,” in *2016 IEEE Spoken Language Technology Workshop (SLT)*, pp. 447–453, IEEE, 2016.
- [44] J. Scheiner, I. Williams, and P. Aleksic, “Voice search language model adaptation using contextual information,” in *2016 IEEE Spoken Language Technology Workshop (SLT)*, pp. 253–257, IEEE, 2016.
- [45] I. Williams, A. Kannan, P. Aleksic, D. Rybach, and T. N. Sainath, “Contextual speech recognition in end-to-end neural network systems using beam search,” in *Proc. of Interspeech*, 2018.

- [46] T. Mikolov and G. Zweig, “Context dependent recurrent neural network language model,” in *2012 IEEE Spoken Language Technology Workshop (SLT)*, pp. 234–239, IEEE, 2012.
- [47] Z. Chen, M. Jain, Y. Wang, M. L. Seltzer, and C. Fuegen, “End-to-end contextual speech recognition using class language models and a token passing decoder,” in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6186–6190, IEEE, 2019.
- [48] J. Cho, M. K. Baskar, R. Li, M. Wiesner, S. H. Mallidi, N. Yalta, M. Karafiat, S. Watanabe, and T. Hori, “Multilingual sequence-to-sequence speech recognition: architecture, transfer learning, and language modeling,” in *2018 IEEE Spoken Language Technology Workshop (SLT)*, pp. 521–527, IEEE, 2018.
- [49] S. Kim, S. Dalmia, and F. Metze, “Situation informed end-to-end asr for chime-5 challenge,” 2018.
- [50] S. Kim and F. Metze, “Dialog-context aware end-to-end speech recognition,” in *2018 IEEE Spoken Language Technology Workshop (SLT)*, pp. 434–440, IEEE, 2018.
- [51] Y. Long, Y. Li, S. Wei, Q. Zhang, and C. Yang, “Large-scale semi-supervised training in deep learning acoustic model for asr,” *IEEE Access*, vol. 7, pp. 133615–133627, 2019.
- [52] Z. Fan, S. Zhou, and B. Xu, “Unsupervised pre-training for sequence to sequence speech recognition,” *arXiv preprint arXiv:1910.12418*, 2019.
- [53] T. Drugman, J. Pylkkonen, and R. Kneser, “Active and semi-supervised learning in asr: Benefits on the acoustic and language models,” *arXiv preprint arXiv:1903.02852*, 2019.
- [54] D. Yu, B. Varadarajan, L. Deng, and A. Acero, “Active learning and semi-supervised learning for speech recognition: A unified framework using the global entropy reduction maximization criterion,” *Computer Speech & Language*, vol. 24, no. 3, pp. 433–444, 2010.
- [55] S. Thomas, M. L. Seltzer, K. Church, and H. Hermansky, “Deep neural network features and semi-supervised training for low resource speech recognition,” in *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6704–6708, IEEE, 2013.
- [56] K. Vesely, M. Hannemann, and L. Burget, “Semi-supervised training of deep neural networks,” in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 267–272, IEEE, 2013.
- [57] Y. Liu and K. Kirchhoff, “Graph-based semi-supervised acoustic modeling in dnn-based speech recognition,” in *2014 IEEE Spoken Language Technology Workshop (SLT)*, pp. 177–182, IEEE, 2014.

- [58] A. K. Dhaka and G. Salvi, "Sparse autoencoder based semi-supervised learning for phone classification with limited annotations," in *Proc. GLU 2017 International Workshop on Grounding Language Understanding*, pp. 22–26, 2017.
- [59] P. Guo, H. Xu, L. Xie, and E. S. Chng, "Study of semi-supervised approaches to improving english-mandarin code-switching speech recognition," *arXiv preprint arXiv:1806.06200*, 2018.
- [60] K. Veselý, C. Segura, I. Szöke, J. Luque, and J. Cernocký, "Lightly supervised vs. semi-supervised training of acoustic model on luxembourgish for low-resource automatic speech recognition," in *Interspeech*, pp. 2883–2887, 2018.
- [61] R. Lileikytė, A. Gorin, L. Lamel, J.-L. Gauvain, and T. Fraga-Silva, "Lithuanian broadcast speech transcription using semi-supervised acoustic model training," *Procedia Computer Science*, vol. 81, pp. 107–113, 2016.
- [62] L. Šmídl, J. Švec, A. Pražák, and J. Trmal, "Semi-supervised training of dnn-based acoustic model for atc speech recognition," in *International Conference on Speech and Computer*, pp. 646–655, Springer, 2018.
- [63] S. Karita, S. Watanabe, T. Iwata, A. Ogawa, and M. Delcroix, "Semi-supervised end-to-end speech recognition," in *Proc. Interspeech*, pp. 2–6, 2018.
- [64] S. Karita, S. Watanabe, T. Iwata, M. Delcroix, A. Ogawa, and T. Nakatani, "Semi-supervised end-to-end speech recognition using text-to-speech and autoencoders," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6166–6170, IEEE, 2019.
- [65] B. Barman, "A contrastive analysis of english and bangla phonemics," *Dhaka University Journal of Linguistics*, vol. 2, no. 4, pp. 19–42, 2009.
- [66] J. Chowdhury, ed., *Adhunik Bangla Ovidhan*. Bangla Academy, Dhaka -1000: Bangla Academy, 2016.
- [67] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [68] T. J. Hazen, "Automatic alignment and error correction of human generated transcripts for long speech recordings," in *Ninth International Conference on Spoken Language Processing*, 2006.
- [69] A. Dev and P. Bansal, "Robust features for noisy speech recognition using mfcc computation from magnitude spectrum of higher order autocorrelation coefficients," *International Journal of Computer Applications*, vol. 10, no. 8, pp. 36–38, 2010.

- [70] S. Ravindran, D. V. Anderson, and M. Slaney, “Improving the noise-robustness of mel-frequency cepstral coefficients for speech processing,” *Reconstruction*, vol. 12, p. 14, 2006.
- [71] J. Patino, H. Delgado, and N. Evans, “The eurecom submission to the first dihard challenge,” in *Proc. INTERSPEECH*, vol. 2018, pp. 2813–2817, 2018.
- [72] J. Patino, H. Delgado, N. Evans, and X. Anguera, “Eurecom submission to the albayzin 2016 speaker diarization evaluation,” *Proc. IberSPEECH*, 2016.
- [73] J. Patino, H. Delgado, R. Yin, H. Bredin, C. Barras, and N. W. Evans, “Odessa at albayzin speaker diarization challenge 2018,” in *IberSPEECH*, pp. 211–215, 2018.
- [74] A. Y. Ng, M. I. Jordan, and Y. Weiss, “On spectral clustering: Analysis and an algorithm,” in *Advances in neural information processing systems*, pp. 849–856, 2002.
- [75] T. Giannakopoulos, “pyaudioanalysis: An open-source python library for audio signal analysis,” *PloS one*, vol. 10, no. 12, 2015.
- [76] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, “Hybrid ctc/attention architecture for end-to-end speech recognition,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.
- [77] T. Hayashi, R. Yamamoto, K. Inoue, T. Yoshimura, S. Watanabe, T. Toda, K. Takeda, Y. Zhang, and X. Tan, “Espnet-tts: Unified, reproducible, and integratable open source end-to-end text-to-speech toolkit,” *arXiv preprint arXiv:1910.10909*, 2019.
- [78] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan, *et al.*, “Natural tts synthesis by conditioning wavenet on mel spectrogram predictions,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4779–4783, IEEE, 2018.
- [79] N. Perraudin, P. Balazs, and P. L. Søndergaard, “A fast griffin-lim algorithm,” in *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 1–4, IEEE, 2013.
- [80] T. Hori, S. Watanabe, Y. Zhang, and W. Chan, “Advances in joint ctc-attention based end-to-end speech recognition with a deep cnn encoder and rnn-lm,” in *Proc. of Interspeech*, 2017.
- [81] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning, “Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora,” in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pp. 248–256, Association for Computational Linguistics, 2009.

-
- [82] L. v. d. Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.
- [83] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, *et al.*, “The kaldı speech recognition toolkit,” tech. rep., IEEE Signal Processing Society, 2011.
- [84] S. S. Shubha, N. Sadeq, S. Ahmed, M. N. Islam, M. A. Adnan, M. Y. A. Khan, and M. Z. Islam, “Customizing grapheme-to-phoneme system for non-trivial transcription problems in bangla language,” 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)*, pp. 3191–3200, 2019.

Appendix A

List of Publications

1. Nafis Sadeq, Nafis Tahmid Chowdhury, Farhan Tanvir Utshaw, Shafayat Ahmed, Muhammad Abdullah Adnan. **Improving End-to-End Bangla Speech Recognition with Semi-supervised Training** Findings of the Association for Computational Linguistics: EMNLP 2020.

Link: <https://www.aclweb.org/anthology/2020.findings-emnlp.169>

2. Nafis Sadeq, Shafayat Ahmed, Sudipta Saha Shubha, Md. Nahidul Islam, Muhammad Abdullah Adnan. **Bangla Voice Command Recognition in End-To-End System using Topic Modeling Based Contextual Rescoring** Proc. of the 45th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2020), Barcelona, Spain, May 4-8, 2020.

ICASSP is the a top conference on Speech and Acoustics with **h5-index 80**

Link: <https://ieeexplore.ieee.org/document/9053970>

3. Shafayat Ahmed*, Nafis Sadeq*, Sudipta Saha Shubha*, Md. Nahidul Islam, Muhammad Abdullah Adnan and Mohammad Zuberul Islam. **Preparation of Bangla Speech Corpus from Publicly Available Audio & Text** Proc. of the 12th International Conference on Language Resources and Evaluation (LREC 2020), Paris, France, May 11-16, 2020.

* Authors contributed equally

Link: <http://www.lrec-conf.org/proceedings/lrec2020/pdf/2020.lrec-1.811.pdf>

LREC is top conference is Computational Linguistics with **h5-index 45**

4. Sudipta Saha Shubha, Nafis Sadeq, Shafayat Ahmed, Md. Nahidul Islam, Muhammad Abdullah Adnan, Md. Yasin Ali Khan and Mohammad Zuberul Islam. **Customizing Grapheme-to-Phoneme System for Non-Trivial Transcription Problems in Bangla Language**. 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) (pp. 3191-3200).

NAACL is top conference in Computational Linguistics with **h5-index 61**

Link: <https://www.aclweb.org/anthology/N19-1322>

Generated using Postgraduate Thesis L^AT_EX Template, Version 1.03. Department of
Computer Science and Engineering, Bangladesh University of Engineering and Technology,
Dhaka, Bangladesh.

This thesis was generated on February 28, 2021 at 12:29am.