

Real Time Bengali Speech to Text Conversion using CMU Sphinx



This thesis was submitted in partial fulfillment of the requirement for the
degree of

Bachelor of Computer Science and Engineering

Under the Supervision of

Dr. Jia Uddin

By

Humayun Kabir (14141004)

Ruhan Ahmed (14101042)

Abdullah Umar Nasib (17341001)

School of Engineering and Computer Science

December 2017

BRAC University, Dhaka, Bangladesh

Declaration

We hereby affirm that the work done for this thesis is based on results attained from our own work. All of the tools and materials used in this text have been properly acknowledged. This thesis, neither in whole nor in part, has been previously submitted to any other University or Institute for the award of any degree or diploma.

Signature of Supervisor

Dr. Jia Uddin

Signature of the Authors

Humayun Kabir (14141004)

Ruhan Ahmed (14101042)

Abdullah Umar Nasib (17341001)

Acknowledgement

We would like to show our immense and sincerest of gratitude to our respected Supervisor Dr. Jia Uddin sir for his support and guidance in this project and the writing of this paper. His constant vigilance on ensuring we always made steady progress and care to always point us in the right path ensured the completion of this Thesis. We are absolutely humbled to have had him as our Supervisor.

We'd also like to thank Almighty Allah for being fortunate enough to remain on track of progress for an entire year both physically and financially without having to face any serious illness or emergencies.

We want to thank our Parents for their constant support as well as our friends who helped in aiding with data collection which was crucial for our research. It wouldn't have been possible to maintain a steady workflow without their help.

Finally, this goes without saying, we want to say a special thank you to BRAC University for enabling us to conduct this research and providing us with the right support and tools to do so.

Table of Contents

Acknowledgement	ii
List of Figures	v
List of Tables.....	vi
List of Abbreviations.....	vii
Abstract	1
Chapter 1: Introduction.....	2
1.1 Motivation	2
1.2 Contribution Summary	3
1.3 Thesis Outline	4
Chapter 2: Literature Review	5
2.1 Detection Files.....	5
2.1.1 Detection Model Flowchart.....	5
2.1.2 Language Model (LM)	7
2.1.3 Acoustic Model (AM)	8
2.1.4 Acoustic Model Generation Flowchart	9
2.1.5 Dictionary	10
2.1.6 Transcript.....	11
2.1.7 File ID.....	12
2.1.8 Phone Set	13
2.2 Transcription to Audio Map	14
2.3 Bengali Language Complexity.....	18
2.4 Existing Works	20
Chapter 3: Proposed Model	22
3.1 System Design.....	22
3.2 Algorithm	23

3.2.1 Unique UTF Word Finder Algorithm.....	24
3.2.2 Bengali UNICODE to Bengali Phonetic English Algorithm	25
3.3 Flowchart.....	28
Chapter 4: Experimental Results.....	29
4.1 Experimental Setup and Data Collection	29
4.1.1 Experimental Setup	29
4.1.2 Data Collection.....	29
4.1.3 Tools Used.....	30
4.2 Experimental Result and Analysis	30
4.2.1 Experimental Result	30
4.2.2 Analysis	32
Chapter 5: Conclusion and Future Work.....	34
5.1 Conclusion.....	34
5.2 Future Work	34
References	35

List of Figures

Figure 1. Vocal to Phone Set Conversion	3
Figure 2. Detection Model Flowchart	6
Figure 3. Acoustic Model Files	8
Figure 4. Acoustic Model Generation Flowchart.....	9
Figure 5. File ID Example.....	13
Figure 6. Phone Set Example	14
Figure 7. Waveform Noise Removal.....	15
Figure 8. Waveform Normalization	16
Figure 9. Syllable Split and Merge.....	17
Figure 10. Word by Word Split.....	17
Figure 11. Bengali Vowels.....	18
Figure 12. Bengali Consonants	18
Figure 13. Conjunct Consonants	19
Figure 14. Bengali Unicode Composition.....	20
Figure 15. Word Mapping.....	22
Figure 16. Vocal Variation Training.....	23
Figure 17. Unique UTF Word Finder Algorithm Flowchart.....	23
Figure 18. Unique UTF Word Finder.....	25
Figure 19. Bengali UNICODE to Bengali Phonetic English Algorithm Flowchart	25
Figure 20. System Training File Generation Flowchart.....	26
Figure 21. OCR Artefacts.....	27
Figure 22. Accuracy Chart	29
Figure 23. Accuracy Calculation Flowchart	29

List of Tables

TABLE I: Sample Language Model	7
TABLE II: Sample Dictionary with Phone Sets	11
TABLE III: Sample Transcript	12
TABLE IV: Result Data	29

List of Abbreviations

AM	Acoustic Model
ASCII	American Standard Code for Information Interchange
CMU	Carnegie Melon University
DAW	Digital Audio Workstation
HMM	Hidden Markov Model
IO	Input-Output
LM	Language Model
MFC	Mel-Frequency Cepstrum
MFCC	Mel-Frequency Cepstral Coefficient
OCR	Optical Character Recognition
PCM	Pulse Code Modulation
PNG	Portable Network Graphics
RIFF	Resource Interchange File Format
SNR	Signal-to-Noise-Ratio
STT	Speech to Text
UTF	Unicode Transformation Format
WER	Word Error Rate

Abstract

This paper aims to demonstrate the use of Speech-to-Text technology to convert Bangla spoken in a natural and continuous state into Bengali UNICODE font with good accuracy. This achievement required the usage of the open sourced framework Sphinx 4 created by Carnegie Melon University (CMU) which was written in Java and provides the required procedural coding tools to develop an acoustic model for a custom language like Bangla. It takes help of algorithms like Baum-Welch to create an Acoustic Model from training data which we gathered ourselves. Our main objective was to ensure that the system was adequately trained on a word by word basis from various speakers so that it could recognize new speakers fluently. We used a free digital audio workstation (DAW) called Audacity to manipulate the collected recording data via techniques like continuous frequency profiling to reduce the Signal-to-Noise-Ratio (SNR), vocal levelling, normalization and syllable splitting as well as merging to ensure an error free 1:1-word mapping of each utterance with its mirror transcription file text. The result is a speech to text recognition system with an acceptable accuracy of around 75% that was trained using recorded speech data from 10 individual speakers consisting of both males and females using custom transcript files that we wrote.

CHAPTER 01

Introduction

The main idea for this project revolves around the conversion of natural speech in the Bengali language into its respective text in UNICODE characters. Speech to text recognition technology has been around for over a decade and speech is the most basic form of human communication. Modern algorithms and processors enable us to extract text data from speech directly using specialized engines and over the years we've been able to optimize all the phases involved to polish the speeds required to achieve this [1]. It has been made possible to do on a smartphone today which was thought to be difficult to do on a desktop computer just years ago [2]. One of the other challenges is to ensure that the recognition worked for a variation of speakers which is difficult to achieve without the proper abstract levels of feature extraction from the data being trained. The specific features to extract from a recording becomes more refined as more data is given to system to learn from for the same set of words being uttered [3, 4]. The framework we used called Sphinx uses the Hidden Markov Model (HMM) [13, 18, 20]. It is seen as the vastly standard method for recognition as it is an efficient algorithm [5]. This design within the framework is different compared to past implementations as it constructs a graph that enables parallelism of decoding at multiple levels. This allows simultaneous feature recognition. The Viterbi algorithm works in conjunction with the Hidden Markov Model to find the most likely sequence of hidden states [21]. This granular abstraction of the different detection phases allows for a more modular and efficient modelling of speech detection.

1.1 Motivation

Bangla is one of the richest languages in the world. Typing in Bangla can be difficult especially for those who cannot type fast. Being able to convert the spoken word to text is the easiest form of typing any language and we want Bangla to be a part of that digital realm by offline means rather using an online service. We have laid the foundation needed to improve the speech detection accuracy which can be used for future work for universal applications.

Bangladesh has masses of people who do not know how to read or write but merely speak, this can be a game changing tool for them as their medium of communication will improve. The

possibilities are endless when a language like Bangla which is used by over 164 million people is fused with speech recognition technology [28]. Figure 1 shows a simplified visual process of the various phases of conversion from speech to waveform to phone sets to “Banglish” which is the phonetic form of Bengali written in English and then finally Bengali text. For example, the Bengali word “আমার” can be written as “Amar” in “Banglish” form.

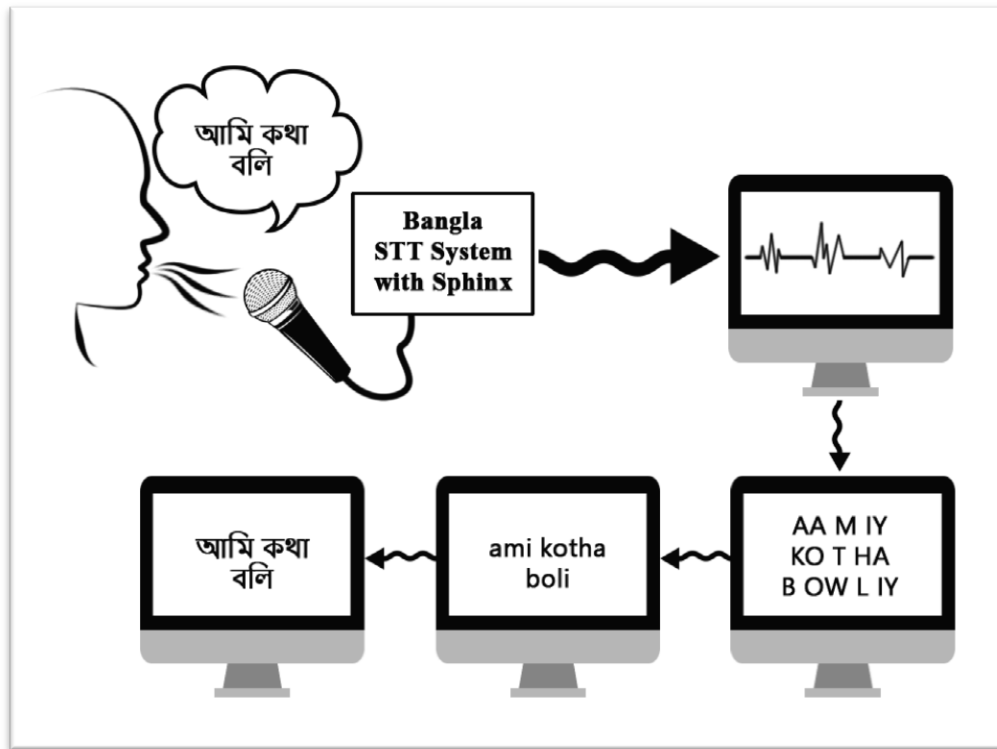


Figure 1. Vocal to Phone Set Conversion

1.2 Contribution Summary

The only research in this arena of speech recognition for Bangla was previously performed with old tools like Microsoft SAPI, Festival and others [4, 14]. At the time of our starting no other such system existed for proper natural and continuous speech recognition in the Bengali language. Thus, we decided to research speech to text recognition for our mother tongue with a more updated approach such that it can be spoken and recognized continuously in real time to match modern processing speeds. Our paper also showcases how the training process can be streamlined using the scripts we wrote to automate many of the process like unique word finding and required training file generation.

1.3 Thesis Outline

- Chapter 2 covers the literature review of the paper which describes the techniques and algorithms applied in this thesis.
- Chapter 3 defines the proposed model including the implementation specifics.
- Chapter 4 displays the experimental results based on collected data sets and their performance comparisons.
- Chapter 5 concludes the paper specifying its difficulties and limitations while stating possible future work for this system.

CHAPTER 02

Literature Review

2.1 Detection Files

The CMU Sphinx framework relies on three key models in order to begin detecting any new language which are the language model (LM), acoustic model (AM) and dictionary file [9, 10]. Each play a key role in the final detection phase of continuous speech recognition in real-time, both in terms of speeding up the detection as well as finding the best accuracy. The main phone sets for any word's syllables are all constructed in English ASCII letters but the underlying words to be recognized need to be in UTF-8 encoding for the language being trained. Speech to text (STT) occurs at the final step when training of the acoustic model files has been completed.

During the training process of the AM (acoustic model), other files are also needed like the transcript, file IDs and phones which are discussed in more details below. These are the basic training building blocks of telling the framework where the audio files are, and what were the words uttered in them. There are quite a few pitfalls in this setup that need to be faced before training commences and numerous error possibilities such as words not aligning to audio or words missing from the unique dictionary. So, we designed specific automation scripts to handle these problems more fluently since the training is an iterative process and needs to be checked and repeated with each iteration before finally obtaining an error free trained model [10].

2.1.1 Detection Model Flowchart

Figure 2 shows the various steps of how the detection model is working starting with voice input which goes onto being dissected at the silence points. Phone sets for each sample are detected and then after cross referencing with the acoustic model, language model and dictionary, the best possible word match is found. This process repeats iteratively until the full duration of the given input waveform has been converted to text. It may require trial and error to get the individual model files right based on the final detection accuracy being represented so during this process, simulations are performed to check the validity of how accurate the detection models are on a word by word basis.

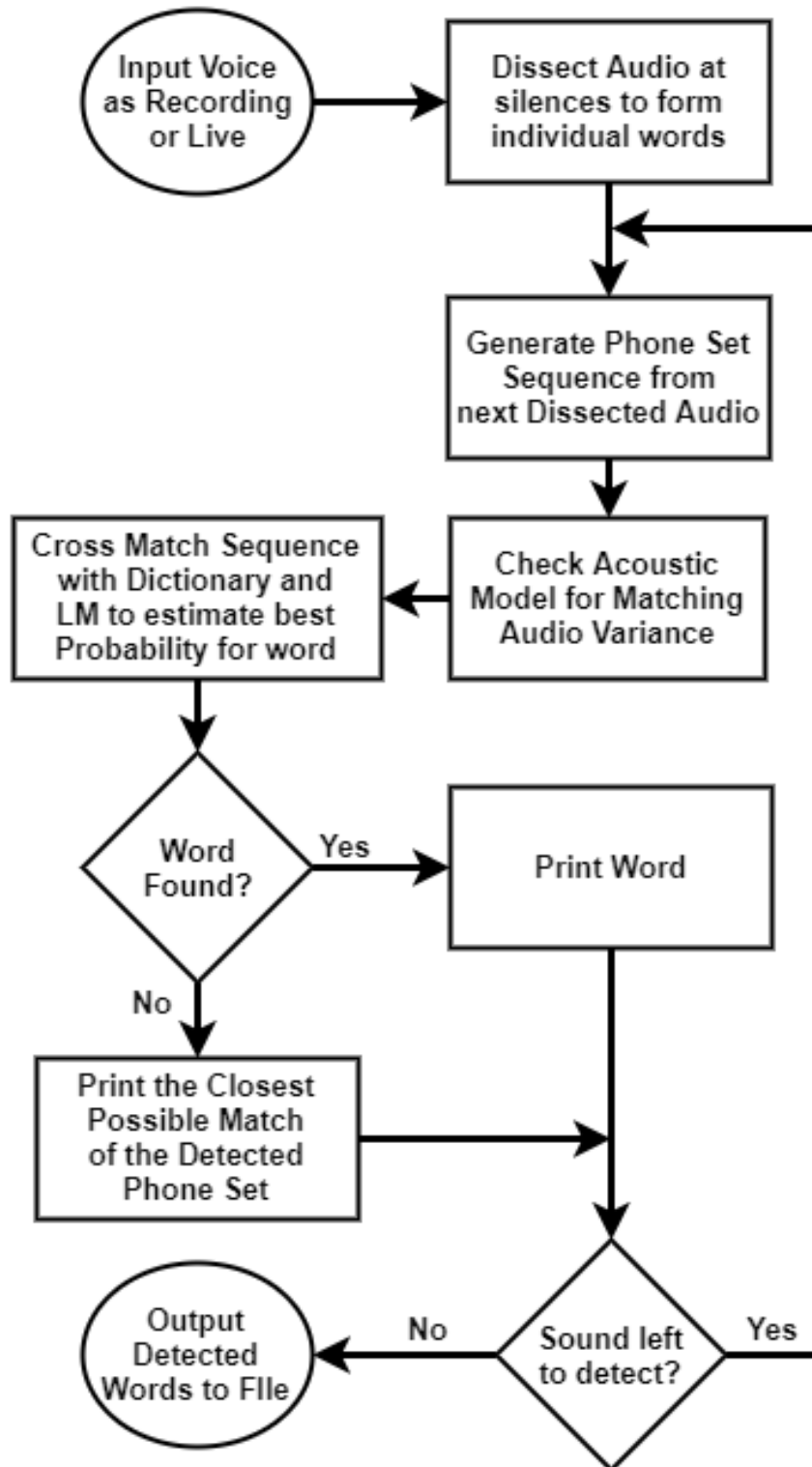


Figure 2. Detection Model Flowchart

2.1.2 Language Model (LM)

The language model (LM) also known as the statistical model contains probabilistic values for what the next word may be given the recognition of the first. It is created using a tool that has rule sets applied to how a sequence of phone sets may align up based on the trained model and this helps it calculate the statistics of the closest occurrence of a series of utterances detected in a given audio file. Table I shows a sample of our language model with its statistical values on the first gram.

The first column shows the probability in terms of logarithm (base 10) of conditional probability 'p' thus giving a negative result. The second column shows the actual word. The probabilities are usually the same for the same first letter in this case 'আ'.

TABLE I: Sample Language Model

Probability	Word
-0.7782	</s>
-0.7782	<s>
-3.4763	আছি
-3.4763	আদরে
-3.4763	আইন
-3.4763	আকাশ
-3.4763	আকাশের
-3.4763	আজ
-3.4763	আলোতে
-3.4763	আলোয়
-3.4763	আলু
-3.4763	আম
-3.4763	আমাদের
-3.4763	আমাকে
-3.4763	আমার

2.1.3 Acoustic Model (AM)

During the training phase of the detection system, a set of files called feature vectors are generated that consist of the Mel-Frequency Cepstral Coefficients (MFCCs) which collectively form the Mel-Frequency Cepstrum (MFC) for storing the binary files. These files work together as the Acoustic Model (AM) which can vary greatly in accuracy depending on the number of hours of recording data given to it. The required files to generate these feature vectors are primarily the audio files as well as some helper files that aide with the training such as the Dictionary, Phones, Transcript, File IDs and Language Model (LM). In order to maximize the IO (Input-Output) efficiency only the static features are stored. The dynamic features however are computed at runtime. Figure 3 shows all of the generated acoustic model files. The first file for instance “feat.params”, as the name suggests, defines the parameters of the features that were extracted during training such as the upper frequency “upperf 6800” or noise filter frequency “nfilt 25”. The file “noisedict” defines the silence tag SIL which is used to avoid silences. The remaining files “mdef”, “means”, “mixture_weights”, “noisedict”, “transition_matrices” and “variances” handle keeping track of statistical probability of phones that appear for specific audio frequencies which are gathered from the training data and stores them in binary format to be read at very fast speeds when detecting real-time speech. They are not human readable.








Name	Type
 feat.params	PARAMS File
 mdef	File
 means	File
 mixture_weights	File
 noisedict	File
 transition_matrices	File
 variances	File

Figure 3. Acoustic Model Files

2.1.4 Acoustic Model Generation Flowchart

Figure 4 shows the various processes by which the acoustic model files are generated from the initial input files like transcript, language model, dictionary, file ID and of course the training WAVE files which all play a role in the training process for feature extraction in order to better detect speech. This training process is very highly error prone as the slightest error in any of the input files can make it terminate and fail the training. The duration needed for training increase with increase of more hours of input data as does the risk of creating errors that become hard to track down.

Thus, it is usually a better idea to dissect and train small durations at a time and slowly add on. This makes debugging a lot simpler. It goes through a series of 7 phases followed by 90 modules of checks in order to determine whether the training succeeded or failed.

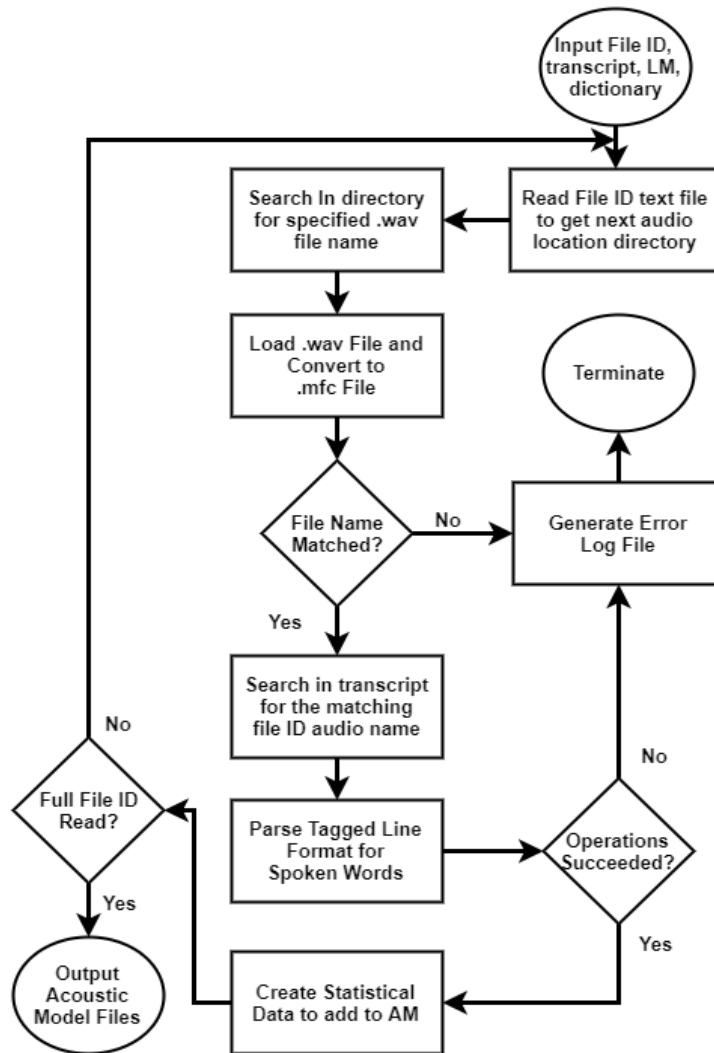


Figure 4. Acoustic Model Generation Flowchart

2.1.5 Dictionary

The dictionary file as the name suggests acts as the main lookup source to translate the detected phone set of an audio sample into its consecutive word in Bangla UNICODE. Sphinx samples the input audio by splitting it at the silences and for each split portion it consults the Acoustic Model to look at past training data and determine what the individual phones may be. After that it cross references the detected phones with the dictionary file to see which sequences best match the word list. It may not always find a 100% match for the phone sequence to word but this is where probability is used from the language model to determine the closest match. An audio sample for continuous speech contains multiple words. So, for example if a stream of phones for 3 individual words are detected, at first it will find a combined sequential phone set for the 3 words but the words aren't known yet. Then the phones are cross matched with the language model and dictionary and the best possible match for the 3 words is printed.

This dictionary file must only contain words that exist in the transcript and vice-versa thus it is very error prone to create manually and also very hectic to get right with each training phase. Thus, we created scripts to automate the process such that no matter what text passage we give as input, it converts them into a transcript as well as a unique word list which are discussed in more details in chapter 3 along with the algorithms we made.

The following table II shows an example of the dictionary file where each new line contains a unique Bengali word followed by a tab in the actual text file and then its respective phone sets sequence separated by spaces. Some words as shown like “আমার” may have multiple variants of phone set sequence. This is because when often uttered, the recorded waveform’s beginning portion for this particular word may sometimes generate the phoneme “AH” or “AA” depending on the breath released after uttering ‘আ’. We have deduced this through trial and error. Listing these variations within the dictionary allows a greater chance for the word to be detected correctly. This is not a hard and fast rule as all words do not apply. Usually it is the more common ones that may need a variant and often it is found through trial and error iteratively. A word can have 3 variations at most as that is supportive limitation of the framework. These are represented as n grams in the Language Model (LM) where ‘n’ is either 1, 2 or 3. The probability values are calculated accordingly for each variation.

TABLE II: Sample Dictionary with Phone Sets

Bengali Word	Phone Set
আছি	AE CH IY
আদরে	AE D AH R
আইন	AH IH N
আকাশ	AE K AH SH
আকাশের	AE K AH SH AH R
আজ	AE JH
আলোতে	AH L OW T
আলোয়	AE L OW IY
আলু	AA L UW
আম	AE M
আমাদের	AH M AE D ER
আমাকে	AH M EY K
আমার	AH M AA R
আমার (2)	AA M A R

2.1.6 Transcript

The transcript file is required during the training phase of the Acoustic model to map the spoken words to its respective audio file. It contains a list of sentences or words or both that define with tags the consecutive audio file where those particular words or sentences were uttered. This file plays a very important part in the final accuracy of the word detection as its 1:1-word mapping with each respective audio training file is crucial to perfecting the Acoustic model needed for accurate phone detection [15, 16]. If a certain sentence has even a single word more or less, the Baum-Welch algorithm will attempt to split the audio provided for that sentence in order to achieve a 1:1 match of audio samples to word count [6, 17, 19]. So, at every stage of training this file requires extensive revisions based on the collected training data. Table III shows a sample transcript with the sentence tags “<s>” similar to HTML with its respective audio file name. These two must match word per word without any errors no matter how minor. For example, if an audio file

contained the recording for the words “আমি গ্রামে আছি” but the transcript contained only “আমি গ্রামে” then this will increase the error rate dramatically. There can even be other cases where the word count is in 1:1 ratio with the audio but the exact word was not uttered. For example, if the recording is “আমি খেলছি”, but the transcript reads “আমি খেলেছি”, then the wrong word is being trained with the wrong sound which immensely decreases the accuracy.

TABLE III: Sample Transcript

Tag	Sentence	Tag	Audio File Name
<s>	বৈশাখ এর ঝড়ে আষাঢ় এর মেঘে শ্রাবণ এর কাশফুলে	</s>	(Ruhan1)
<s>	চার রাস্তার মোড়ে পার্কে দোকানে শহর গ্রামে এখানে	</s>	(Ruhan2)
<s>	আম জাম কলা আঙ্গুর কাঠাল জাম্বুরা আপেল পেয়ারা	</s>	(Torab1)
<s>	তেতুল পানি পান দই দুধ লেবু শশা কাচ্চি বিরিয়ানি	</s>	(Torab2)
<s>	ডাল	</s>	(Nasib1w1)
<s>	পটল ভাজি ডিম সিদ্ধ কাচা চেয়ার টেবিল মোবাইল	</s>	(Nasib2)
<s>	জামা কাপড় প্রথমত দ্বিতীয়ত তৃতীয়ত শেষ পর্যন্ত	</s>	(Nasib3)

2.1.7 File ID

This file handles keeping track of all the audio files being used for training and their respective directories inside the training folder. The transcript file uses this to map its sentences with the provided audio WAVE file. During the training process it is also used to keep track of all the files to convert into a special format called MFC (Mel-Frequency Cepstrum) which represents the short-term power spectrum of a sound that is required for training efficiency. The initial files need to all be in a specific format of RIFF (Resource Interchange File Format) data stored as a Microsoft PCM (Pulse Code Modulation) WAVE audio file with parameters of 16-bit, mono and 16,000 Hertz [5]. Figure 5 shows an example of the folder structure on the left side and the file ID text file on the right. Files like “arafe1” and “arafe2” are the “.wav” files within its respective folder. The file “sil.mfc” is basically a binary file containing a few seconds of silence such that the system can recognize it easily. In this manner every WAVE file must be included here. In our model B of data collection, we split up every sentence uttered into individual “.wav” files per word and this file ID text file had to keep track of each one as did the transcript. We automated the process via scripting.

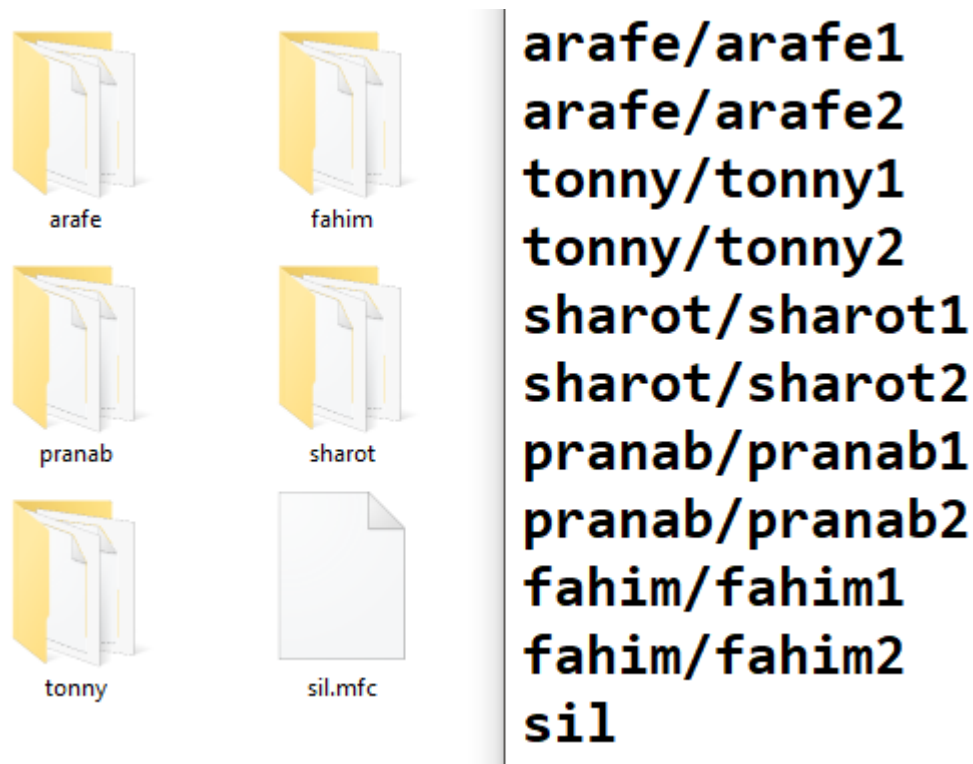


Figure 5. File ID Example

2.1.8 Phone Set

This is a file containing a list of every unique phone used for every word in the dictionary. These phones are the fundamental building blocks of syllables of words in a recording. They are written one per line, often as either two characters or a single character and their combination form the pronunciation of the full word. There's a special phone called SIL present no matter which language is required to be detected as it represents the silenced parts of an audio recording. This is also a very fundamental file to get right as it drastically affects the final accuracy of detection. It is not always clear what the individual phones of a word may sound like in broken form and sometimes may not be possible to be automated by the dictionary tool as it can lead to very bad WER (Word Error Rate). So, often the correct conversion of words to phonemes can be an iterative process with certain languages, especially ones as complex as Bangla which contains symbols used as conjunctions with base characters in order to modify the utterance [7, 8]. There can also be conjunctions of two base characters which are known as "juktakkhor" in Bangla that make the conversion even trickier but, with fine tuning the result, this error is reduced.

As figure 6 shows, our file contains all of the phones we found to be relevant amongst all the vowels and consonants present in the Bengali language. Some of them aren't necessary for computation during utterance so were left blank whereas others are similar or can even be a compound of more than 1 phone. The 39 consonants and 11 vowels sum up to 60 phones as shown in figure 6.

1	অ	OW	16	ঐ	OI	31	ঞ	NG	46	ম	M
2	আ	AA	17	ঐ	OI	32	ট	T	47	য	J
3	া	AA	18	ও	OW	33	ঠ	TH	48	র	R
4	ই	IY	19	ো	OW	34	ড	D	49	ল	L
5	ি	IY	20	ঔ	OW OI	35	ঢ	DH	50	শ	SH
6	ঈ	IY	21	ৌ	OW OI	36	ণ	N	51	ষ	SH
7	ী	IY	22	ক	K	37	ত	T	52	স	SH
8	উ	UW	23	খ	KH	38	থ	TH	53	হ	H
9	ূ	UW	24	গ	G	39	দ	D	54	ড়	R
10	ঊ	UW	25	ঘ	GH	40	ধ	DH	55	ঢ়	R
11	ূ	UW	26	ঙ	NG	41	ন	N	56	য়	Y
12	ঋ	R IY	27	চ	C	42	প	P	57	ৎ	T
13	ৄ	R IY	28	ছ	CH	43	ফ	F	58	ং	NG
14	এ	EY	29	জ	J	44	ব	B	59	ঃ	-
15	ে	EY	30	ঝ	JH	45	ভ	V	60	ঁ	-

Figure 6. Phone Set Example

2.2 Transcription to Audio Map

In order to ensure that the transcript mapped 1:1 with every transcript line, some manual editing steps needed to be performed on every single audio file to ensure it was as error free as possible. The first objective for each audio file was to remove the background noise from each of the received recordings from our volunteer speakers which involved sampling a continuous recognizable frequency of noise and then profiling it to remove it from the vocals. Sometimes exception frequencies like random car horns or people in the background came up and those needed to be dealt with manually as well.

Figure 7 shows a constant frequency portion of the audio on the top before the first word it uttered is profiled in order to clean the entire audio as shown at the bottom. All noises do not disappear as some frequencies do not follow a constant noise pattern and need to be handled manually. Thus, the silences between every two sets of utterance are selected as accurately as possible and then set to 0 amplitude in order to get a clean visual track of only the words uttered and nothing else. This step makes it simple to break up the entire audio file into multiple “.wav” files

containing one word each with at least 0.2 seconds of silence before and after each word.

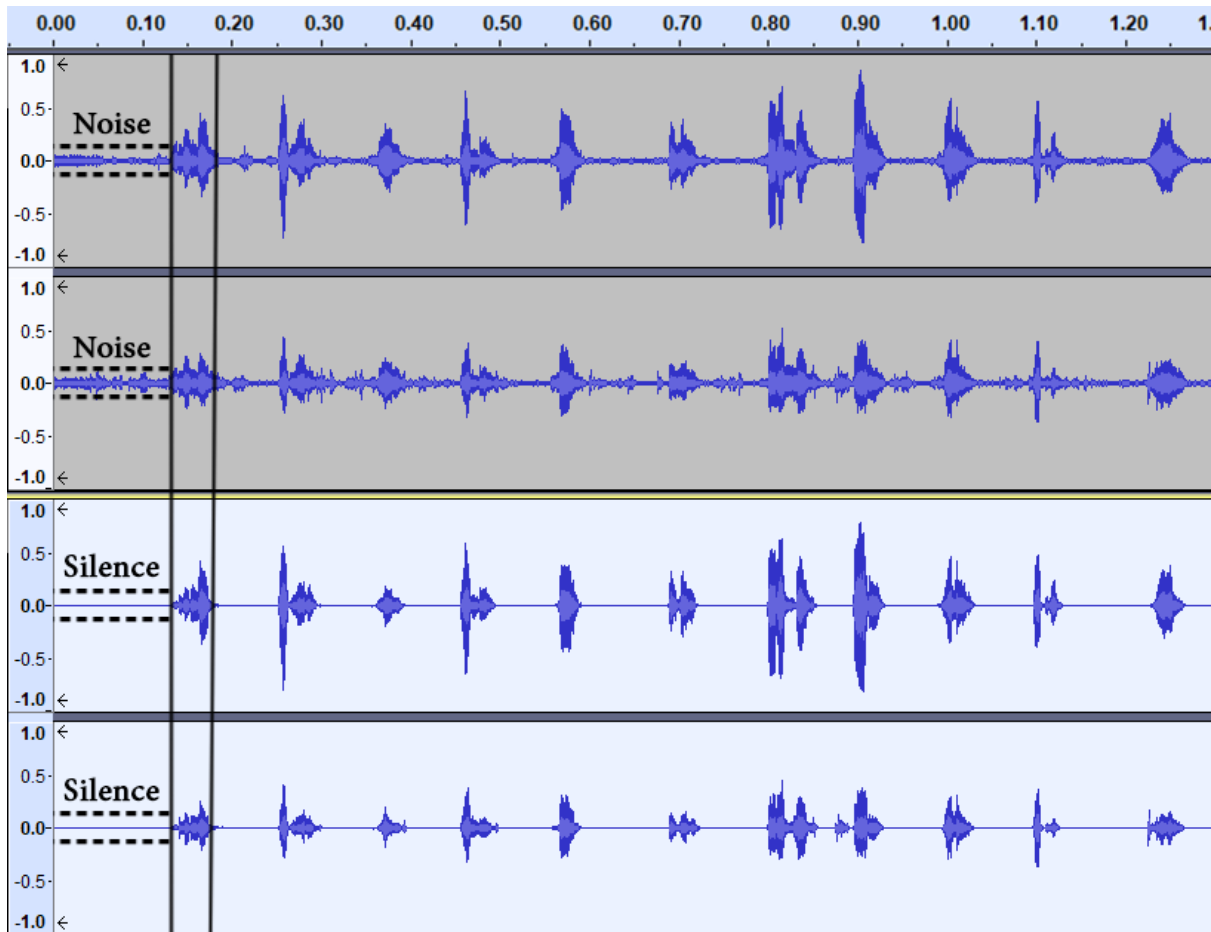


Figure 7. Waveform Noise Removal

The next step was to normalize the vocals which involved boosting the low frequencies while ensuring that the high ones were not over amplified. There are automated approaches to this with our DAW Audacity but more often we needed to check on a word by word basis on both left and right channels to ensure the highest possible quality for each uttered word [22]. Since the system only accepts mono channel audio, we either selected the best one of the two available or merged them into one.

As figure 8 shows, the top two waveforms represent the left and right channels of the same audio recording and each wave differs vastly based on the distance of the speaker from the channel splitter of the microphone. During normalization certain words may peak more than others based on the max neighboring amplitude. This is evened out through equalization and compression techniques.

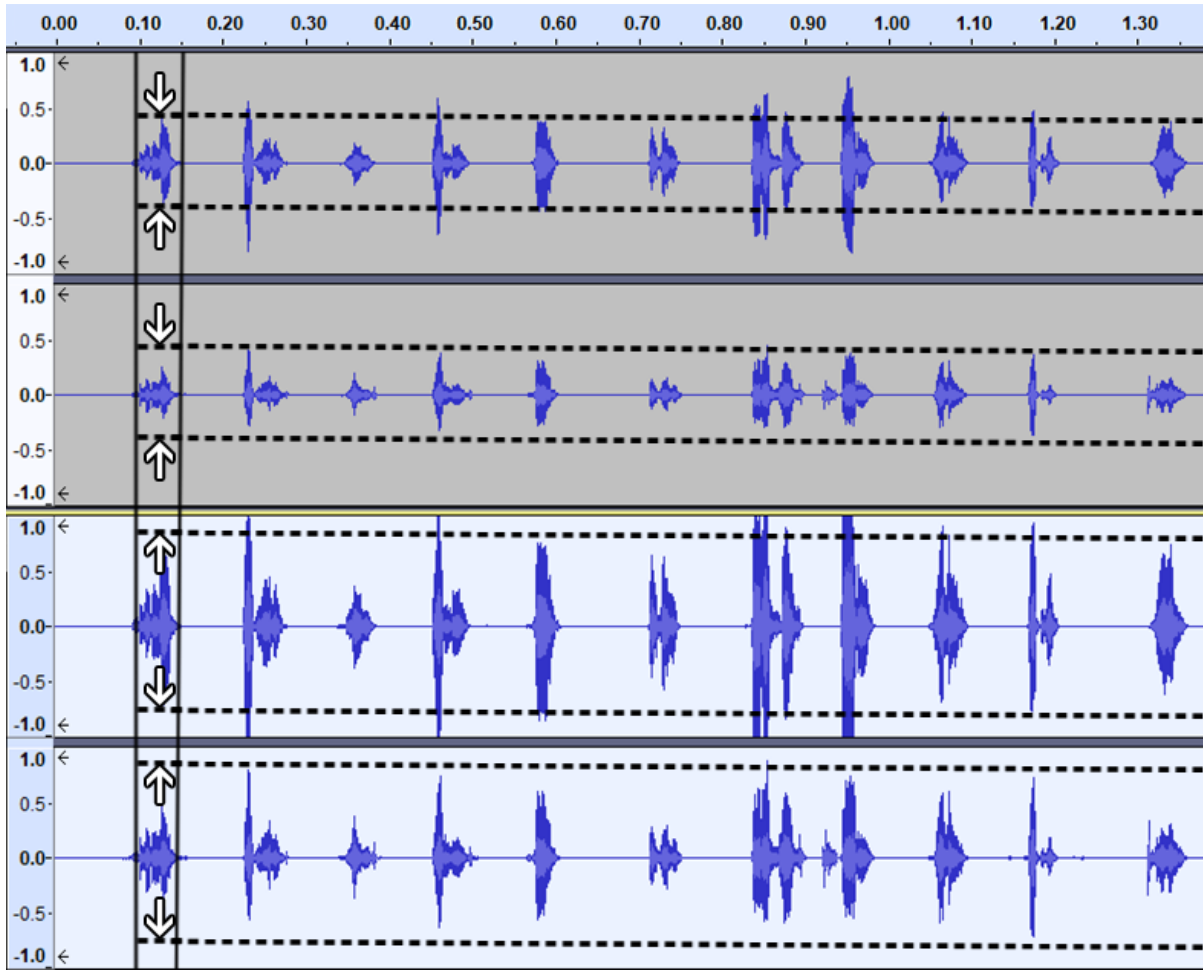


Figure 8. Waveform Normalization

Sometimes certain words that contained multiple syllables, were uttered with a small silence in the middle whereas other words were spoken too continuously that it ended up as a merged single waveform without a silent gap in the middle. During the training process the words needed to be split and a singular waveform for each utterance so that the system could recognize and map each audio partition to its respective transcript word. So, sometimes we had to zoom in very close to see the fine granularity of the wave shifts to determine the best positions to split or merge a word at because if this was done poorly, the neighbouring syllables might have been affected which would damage the resulting sound and hence the training procedure. Figure 9 displays the merging and splitting of words that were too separate or too close respectively.

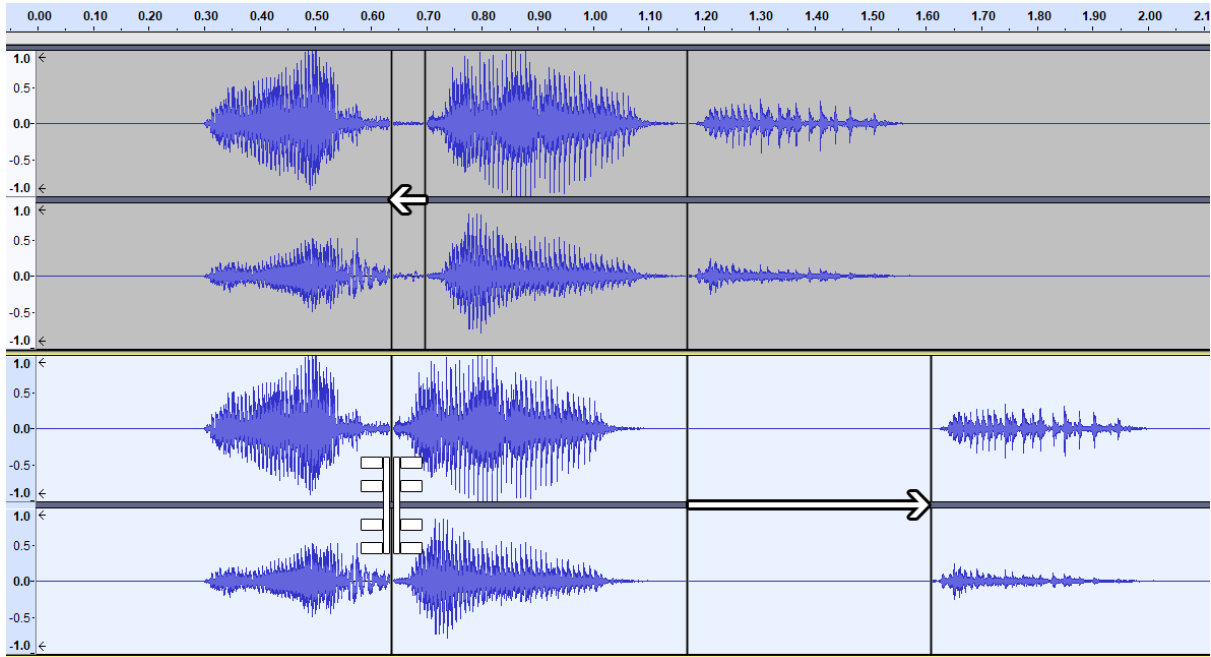


Figure 9. Syllable Split and Merge

Finally, after each word was cleaned, normalized and syllable-wise corrected, they were required to be split up into separate WAVE files and this portion could be automated by fine tuning the settings to automatically detect words at a set threshold value of silence apart. We also needed to ensure at this stage that each word selected area had at least 0.2 seconds of silence before and after it otherwise it would cause training issues if the waveform was too compacted.

Figure 10 shows how the label layer is separate and synced perfectly with each individual waveform while ensuring about 0.2 seconds of silence before and after. The labels are numbered which makes exporting easier with the prefix being the name of the speaker and the sentence number.

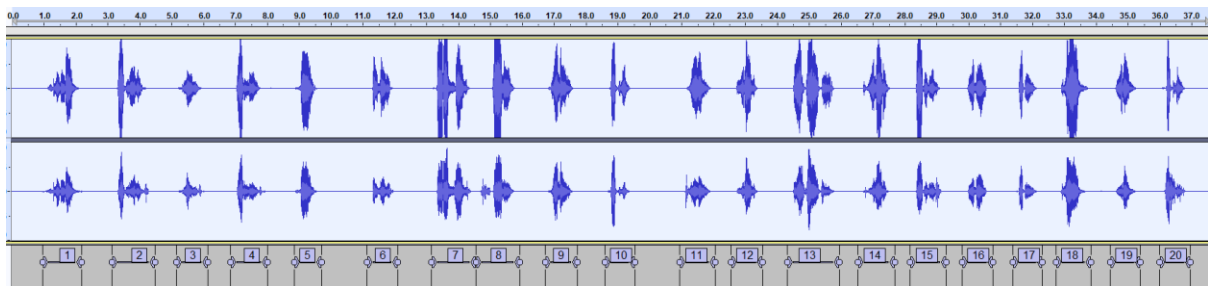


Figure 10. Word by Word Split

2.3 Bengali Language Complexity

The Bangla language has various degrees of complexity when it comes to writing but this is due to the fact that it is a very phonetically worded language where each word is spelt as it should sound like, unlike languages like English where pronunciations can vary greatly from the spelling [26]. Thus, its construct in digital form represents a larger challenge.

Figure 11 shows 11 vowels or “shorborno” in Bengali which are “অ, আ, ই, ঈ, উ, ঊ, ঋ, ঌ, ঍, ঎, ও, ঔ” and also the consonant ‘ক’ in its diacritic form of the vowels.

অ	আ	ই	ঈ	উ	ঊ	ঋ	ঌ
a	ā	i	ī	u	ū	ṛ	e
ঐ	ও	ঔ	ক	কা	কি	কী	কু
ai	o	au	ka	kā	ki	kī	ku
কু	কৃ	কে	কৈ	কো	কৌ		
kū	kr	ke	kai	ko	kau		

Figure 11. Bengali Vowels

ক	ka	[kɔ]	খ	kha	[kʰɔ]	গ	ga	[gɔ]	ঘ	gha	[gʰɔ]
ঙ	ña	[ŋɔ]	চ	ca	[tʃɔ]	ছ	cha	[tʃʰɔ]	জ	ja	[dʒɔ]
ঝ	jha	[dʒʰɔ]	ঞ	ña	[ɲɔ]	ট	ṭa	[ʈɔ]	ঠ	ṭha	[ʈʰɔ]
ড	ḍa	[ɖɔ]	ঢ	ḍha	[ɖʰɔ]	ণ	ṇa	[ɳɔ]	ত	ta	[tɔ]
থ	tha	[tʰɔ]	দ	da	[dɔ]	ধ	dha	[dʰɔ]	ন	na	[nɔ]
প	pa	[pɔ]	ফ	pha	[pʰɔ]	ব	ba	[bɔ]	ভ	bha	[bʰɔ]
ম	ma	[mɔ]	য	ya	[dʒɔ]	র	ra	[rɔ]	ল	la	[lɔ]
শ	śa	[ʃɔ/sɔ]	ষ	ṣa	[ʃɔ]	স	sa	[sɔ/ʃɔ]	হ	ha	[ɦɔ]
য়	ya	[jɔ]	ড়	ṛa	[rɔ]	ঢ়	ṛha	[rʰɔ]			

Figure 12. Bengali Consonants

Figure 12 shows all the consonants in the Bengali language. But there's another way to form special consonants from these base ones through conjunction. Bengali conjunct consonants are a form of fusing two or more base consonants to form a letter called "juktakhor" which literally means combined character in Bengali and this enables a powerful way to form new syllables to form more complex words that sound exactly as spelt. Figure 13 shows some examples of the conjunct forms of a few consonants. There can be many more.

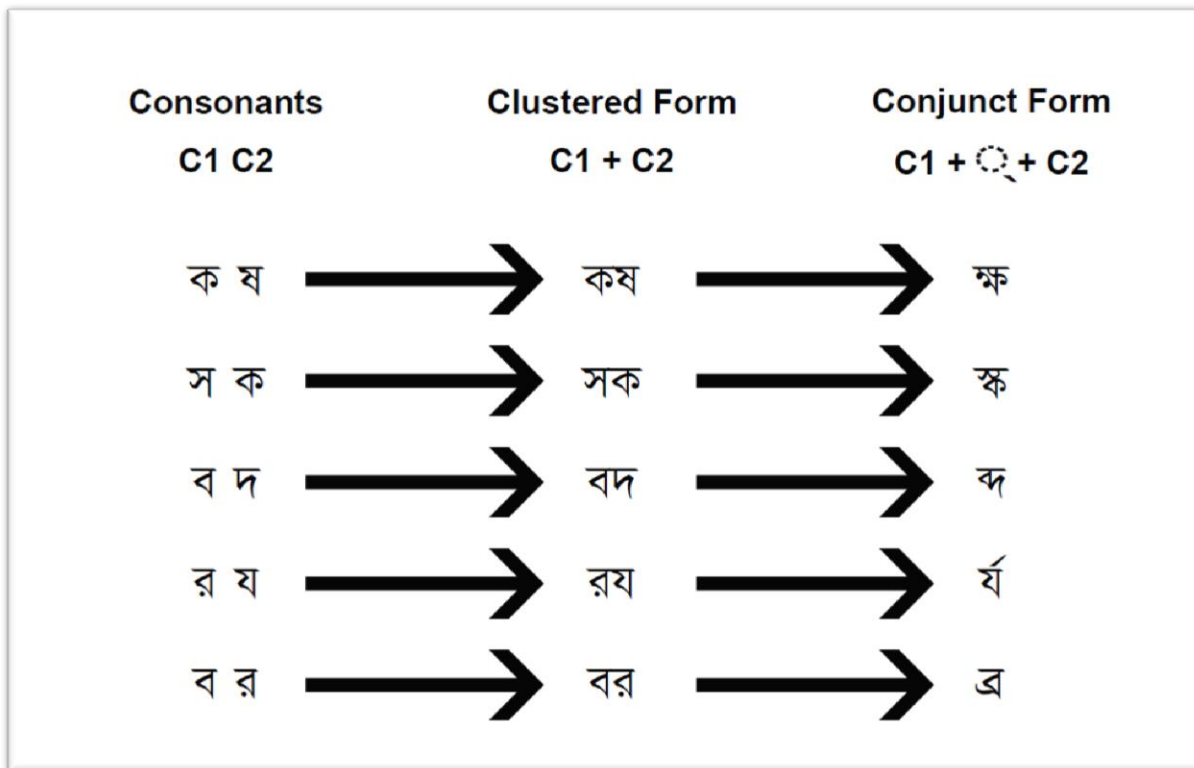


Figure 13. Conjunct Consonants

Bengali UNICODE is often composed together with the diacritic form of the vowels as shown in figure 14 in order to get a modified sound of the original consonant. This also helps create new syllables that affect words in different ways which pose computational problems as well. In code once the characters are composed, they become a single character and even small variances in the decomposed form may lead to a single character still appearing the same but being of different UNICODE. This problem was precisely handled with our unique UTF word finding algorithm.

Decomposed Form	Unicode of Decomposed Form	Use with a Consonant	Equivalent Composed Form	Unicode of Composed Form
ে া	09CB 09BE ে া	ক ে া ↓ ↓ = কো	ো	09CB ো
ে ী	09C7 09D7 ে ী	ক ে ী ↓ ↓ = কৌ	ৌ	09CC ৌ

Figure 14. Bengali UNICODE Composition

2.4 Existing Works

Speech to text for any language is the process of dissecting discrete syllables or phonemes of recorded vocal audio and converting them to their literal transliteration [11]. Then it's just a matter of cross referencing the converted text which is usually in English letters with an acoustic model dictionary of the custom language to decipher its respective text, usually in its native UNICODE font.

Contributions in this exact field were previously achieved using old tools like Microsoft SAPI but it was very much limited in being slow in recognizing continuous speech without proper word gap [4]. We hope to achieve faster performance using Sphinx 4.

Speech recognition has vast fields of applications, the most beneficial today being automation of smart phone or household tasks via virtual assistants. This is already available worldwide in multiple languages via virtual assistants like Siri on the iPhone, Google Now on Android or Cortana on Windows. Bengali unfortunately has been missing from this arena but with our research helping to bring speech to text for this language, it allows for a future possibility of an assistant that can understand spoken commands in Bengali.

The purpose of our approach is to use a modern engine like the CMU Sphinx 4 that allows far more functionality than old engines like SAPI to deliver faster continuous speech recognition via improved algorithms [24, 25]. If the language takes a more modern approach than previous older technology then it will be possible for implementation into mobile phones as part of a virtual assistant or even an app.

The reason that the Bengali language has fallen behind in this regard is because it is a very tricky language with a lot of challenges to overcome to do correctly. First and foremost, the Bengali language is one of the most accurate in terms of pronunciation based on how each word is written compared to any other language because each letter has numerous variations that can be done to achieve almost any sort of word desired. Thus, computationally this becomes a challenge in terms of developing the phonetic dictionary which is used to match portions of the audio to their respective phonemes that get jointed into syllables to finally form the desired word [12]. As Sphinx was designed for English based letters our key approach was to convert the spoken Bengali speech into the modern widely renowned “Banglish” form which is essentially Bengali written with English letters. Then it’s just a simple matter of using something like the Avro’s conversion chart to decode the Banglish syllables into the final Bengali UNICODE text form.

CHAPTER 03

Proposed Model

3.1 System Design

In our proposed technique, we can divide the whole system into two different segments named ‘Training’ and ‘Testing’. Firstly, in the training segment, we train our system using a number of modules. The technique takes text input using two methods. One is ‘Transcript’ which is shown in table IV and ‘file name tagged text’ is the other which is in the format “<<<<SpeakerName>>>> Text <<AudioName>>>” which parses an input text to collect the respected speaker name and audio name for the spoken text to train. We interchanged between the two based on our needs. The way we input transcripts is, the tags such as <s> ডাল </s> (Nasib1w1) where “ডাল” is tagged with the speaker name ‘Nasib’, script line number 1 and word number 1 gets removed and stored in this phase. Before that, when we took the audio input as data set for training we had to divide the vocal recordings into words. Therefore, splitting the wave in parts to train the system was the first phase. The following figure 15 shows how a sentence 5 words may appear in waveform.



Figure 15. Word Mapping

We trained using 10 different speakers both male and female so that several variations of the same uttered word existed in the system. Each word can vary vastly based on the speaker’s vocal range and this can easily be seen in the following diagram.



Figure 16. Vocal Variation Training

As shown in figure 16, the nature of the waveforms varies greatly for the same two uttered words “আমি” and “বাংলায়”. Differences in pitch and timing are precisely what the system needs to build a good model. By training with multiple variations of vocals, we not only eliminate errors but also help the system recognize new speaker’s voices who weren’t involved in the training process which is essentially the goal [23].

3.2 Algorithm

We created two algorithms to correct word spellings that appear the same but are vastly different in order to obtain the most unique form of each word as well as their Banglish equivalent in order to generate the phone sets.

3.2.1 Unique UTF Word Finder Algorithm

```
List of Bengali Tokens is input that contains duplicates
For Each Bengali Token
    Convert to Bengali Phonetic English using algorithm in Section 3.2.2
    Add to Banglish Token List
End
Create Unique and Duplicate Hash Sets //Hash Sets do not contain duplicates
For Each Banglish Token
    If it does not exist in Unique List
        Add to Banglish Unique List
    Else
        Add to Banglish Duplicate List
End
For Each Banglish Duplicate
    Create new Lines Integer List //Stores the lines with duplicates
    For Each Banglish Token
        If Banglish Token Equals Banglish Duplicate
            Add iteration value to Lines List
        End
    Create new Bengali Duplicate List //Stores all duplicates of same word
    For Each Lines value
        Get Bengali Tokens value at position [Lines value]
        Add the returned value to Bengali Duplicate List
    End
    For Each Bengali Token
        For Each Bengali Duplicate
            If Bengali Token Equals Bengali Duplicate
                Set this Bengali Token to first value of Bengali Duplicate
            End
        End
    End
End
Return new Bengali Tokens List //No Duplicate Entries
```


The following figure 17 shows the unique UTF word finder algorithm in flowchart form which uses various nested loops and lists in order to determine which words are duplicates and replaces them with a single unique copy.

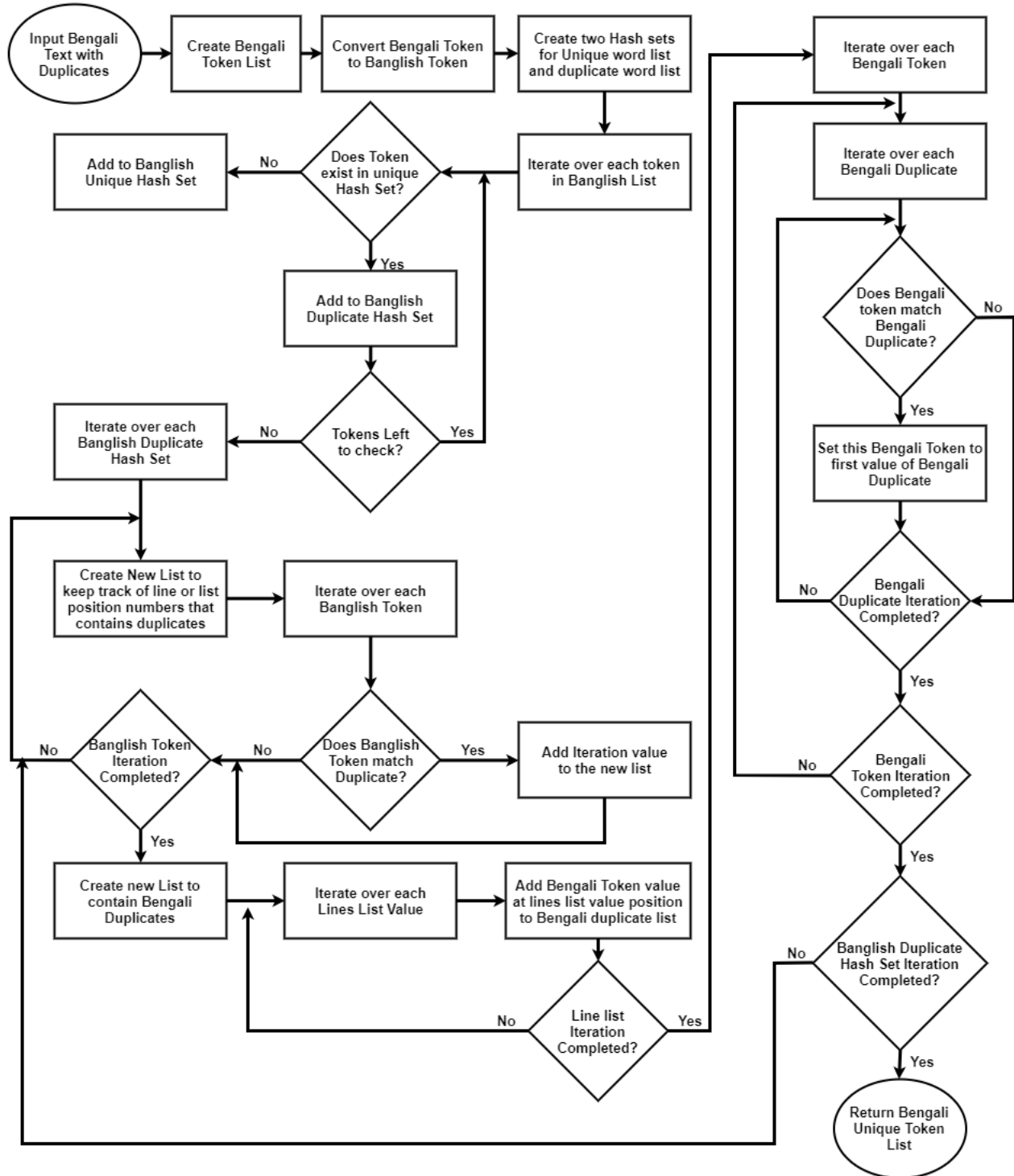


Figure 17. Unique UTF Word Finder Algorithm Flowchart

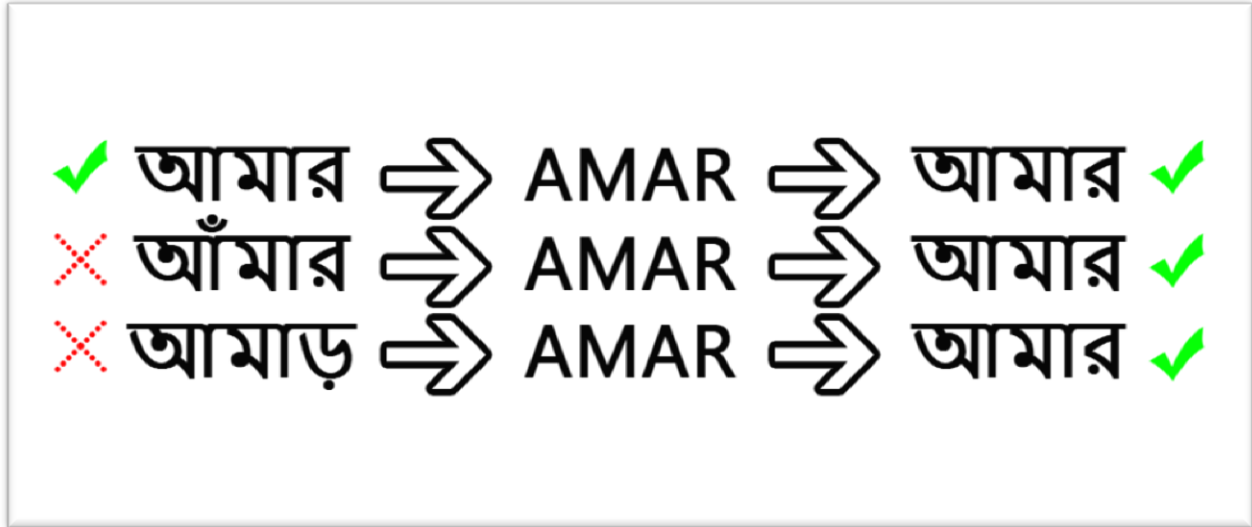


Figure 18. Unique UTF Word Finder

As figure 18 shows, the unique UTF word finding algorithm corrects the spelling mistakes by converting them to “Banglish” to unify the spelling and then replacing them with the correct version. This is performed iteratively for every variation of every word to only have a single unique spelling for each at the end.

3.2.2 Bengali UNICODE to Bengali Phonetic English Algorithm

Create Bengali Token and Banglish Token List

For Each Bengali Token

Remove Special Characters that are not needed

Fix Special Exceptions in Certain UNICODE Characters

Cross Reference Character Sequence with Stored Phonetic English Equivalent

Replace Matching Character Sequences

Add to Banglish Token

End

Return Banglish Token

The following figure 19 shows a flowchart form of this algorithm where the Bengali text is input which is turned into a token list. A separate “Banglish” token list is also created where each respective Bengali word is added after conversion. Then the final token list is returned as output.

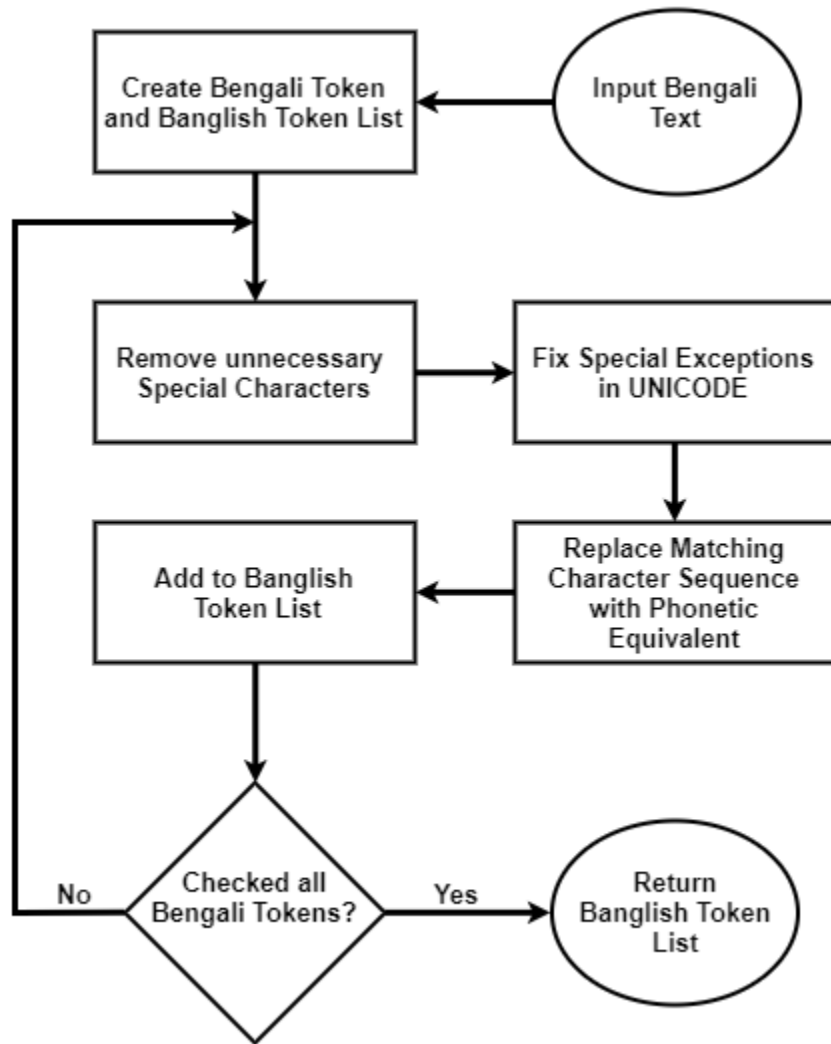


Figure 19. Bengali UNICODE to Bengali Phonetic English Algorithm Flowchart

The following flowchart in figure 20 shows how different types of input are used by our system to generate the required files for the detection models. The LM however is generated by a separate tool which is specific to the Sphinx framework but the rest of the files are created by our own automated scripts.

3.3 Flowchart

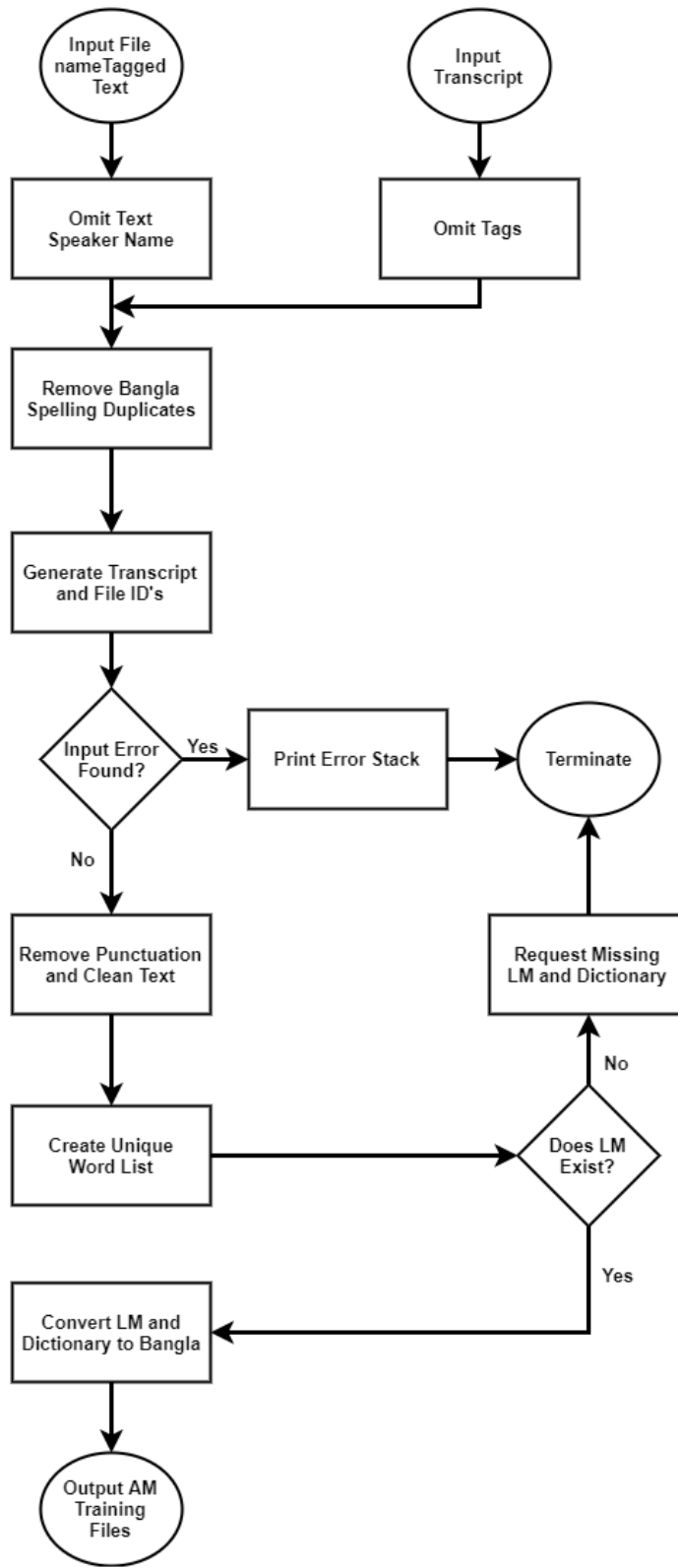


Figure 20. System Training File Generation Flowchart

CHAPTER 04

Experimental Results

4.1 Experimental Setup and Data Collection

4.1.1 Experimental Setup

The experimental setup involved starting with a single speaker to learn the training process and then moving up to multiple speakers to see how the accuracy levels changed. We used various methods of data collection techniques to see how we could improve the final output to increase accuracy. Some proved more useful than others as discussed below.

4.1.2 Data Collection

The first method of data collection that we attempted was to gather audio recordings of Bengali books as well as their text file to create the transcript. Unfortunately, the Bengali books were only available in scanned image form. So, we needed to find a quick way to convert the full books into text and for this purpose we used a separate Optical Character Recognition (OCR) system to convert the extracted image files from PNG (Portable Network Graphics) format to text [27]. But this technique of collection proposed various difficulties one of which was the fact that the images had artefacts as a result of scanning and these caused poor recognition of text in the image. This caused a lot of incorrect spellings to occur as shown in figure 21 where a blurred mark caused an incorrect spelling for the word “আমার” to be generated which shouldn’t have the symbol ‘ঁ’.

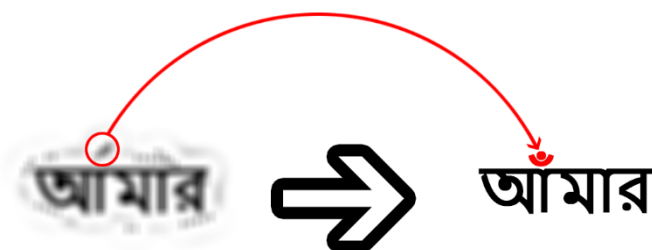


Figure 21. OCR Artefacts

The book recordings also posed problems in the form of too fast reading. The speech wasn't in natural form as it was too fast paced and this caused problems for the Baum-Welch algorithm to properly detect which part of the waveform maps to which word in the transcript. This wasn't a good first approach to train the model as it gave very poor accuracy. But it did allow us to learn the process by which the training method worked and it also gave us an insight into which parts of the training we could automate by scripting in order to improve our workflow to get faster iterations of simulating our models to determine their performance changes.

The next methods involved creating our own written transcripts which we then perfected to have as many unique words as possible. We gathered volunteers willing to lend some time and their voices to make recordings out of the transcripts we wrote. The recording data that they provided had errors such as words missing in the audio file or different words uttered than what was written in the transcript. We still attempted to train this data by performing manual corrections. So, we divided each of their recorded wave files into individual words and also change the transcript accordingly. We manually checked for errors until the complete data set was ready.

4.1.3 Tools Used

We used various sets of tools depending on our needs like the DAW called Audacity for audio correction and manipulation. We required the use of a software called "Bulk Rename Utility" to be able to quickly rename multiple audio files based on our needs which offered the use of regular expression based editing. We primarily used the Eclipse integrated development environment to write our scripts and also used text editors like Notepad++ and Sublime text to edit the UNICODE text in the files. We also used two separate modules of the framework Sphinx, the "sphinx train" module for training the system and the "Sphinx 4" detection module for simulating a live data feed and measuring the performance.

4.2 Experimental Result and Analysis

4.2.1 Experimental Result

The following experimental results were obtained via our simulations as shown in table IV in table form and also in figure 22 in a chart form. As we increased the number of speakers and hours the accuracy gradually increased and perhaps further increase up to 50 hours of data would've enabled us to get the best accuracy.

TABLE IV: Result Data

Model	Training		Testing	
	Number of Speakers	Training Hours	Number of Speakers	Accuracy
A	2	3.0	1	35%
B	5	7.0	2	53%
C	7	9.0	2	70%
D	10	12.0	10	75%

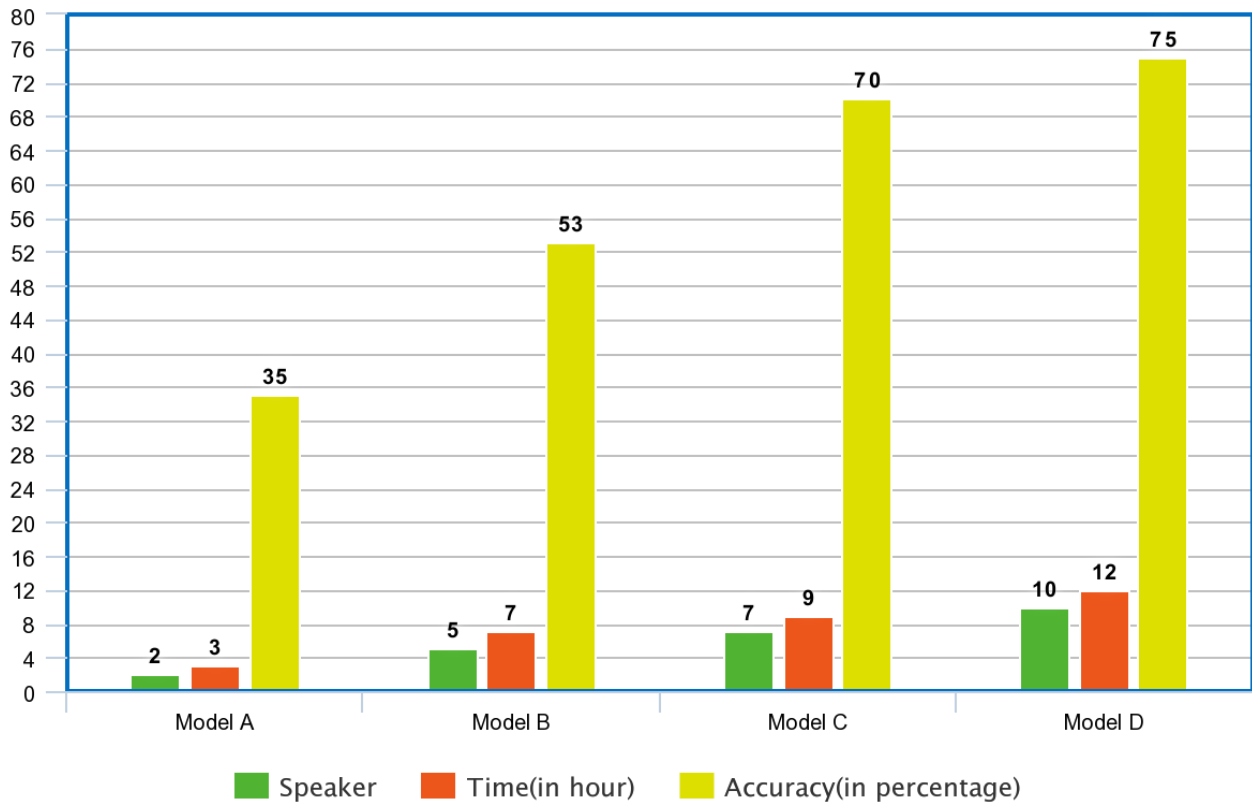


Figure 22. Accuracy Chart

1. Model A: Dataset of 2 male speakers containing 3 hours and 380 unique words.
2. Model B: Dataset of 5 speakers containing 7 hours and 503 unique words of which four were males and one was female.
3. Model C: Dataset of 7 speakers containing 9 hours and 503 unique words of which six were males and one was female.
4. Model D: Dataset of 10 speakers containing 12 hours and 553 unique words of which eight were males and two were females.

For our machine configuration, we used an Intel Core i5-4200M model processor with 4 CPUs clocked at 2.50 Gigahertz each with an Intel HD integrated graphics processor. For memory we used 8 Gigabytes of Random Access Memory running on Windows 10 professional 64-bit.

4.2.2 Analysis

As shown in table IV and figure 22, in experimenting with the book recordings in Model A, the accuracy rate was very poor, 35% to be exact. Testing with Model B containing the voices of four male and one female speaker of 7 hours data set gave a better accuracy. The experimental result was 53% which was much better than Model A as shown in Figure 22. But still the low amount of data was decreasing the accuracy rate. using our third dataset Model C, having 503 words of 7 speakers and almost 9 hours in length, we managed to get an accuracy of 70%. Then our final dataset which is Model D gives the best accuracy rate of 75%. Our analysis shows that increasing number of speakers and hours does increase accuracy but it also has a limitation.

People all over the world pronounce the same word differently so it stands to reason that if we could train our system with every spoken variation for each word, it could definitely work much more accurately. Ideally, we would want 200 different speakers with around 10 minutes of data per speaker to get the optimum accuracy which the sphinx 4 documentation says would be the optimum amount for a well-trained model [5]. We couldn't obtain such a large amount of data but based on our result a greater variety of speakers would definitely improve accuracy [4].

We used a Java program to calculate the accuracy of our models which checks if our given input word is 1:1 mapped with our output word and also if the content of each word matches or not. While testing, we knew about the content of the sentences uttered in the recorded wave files. We wrote it down in a text file line by line with its respective audio file name tagged at the end of each sentence. Then we ran our system to go through the entire list of 300 audio files individually while saving each detection text output in a different text file. It was named with the file name tag at the end of each line mentioned earlier. After we finished detecting all the files, we ran our accuracy test script to check the accuracy. This script matched each word from the audio detection text file with each word that we wrote manually knowing what was actually uttered in the audio. If any word was found to not match, we incremented the error rate variable. After the process completes, we output a single percentage of correct words detected to total number of words. The following figure 23 shows this entire process in the form of a flowchart.

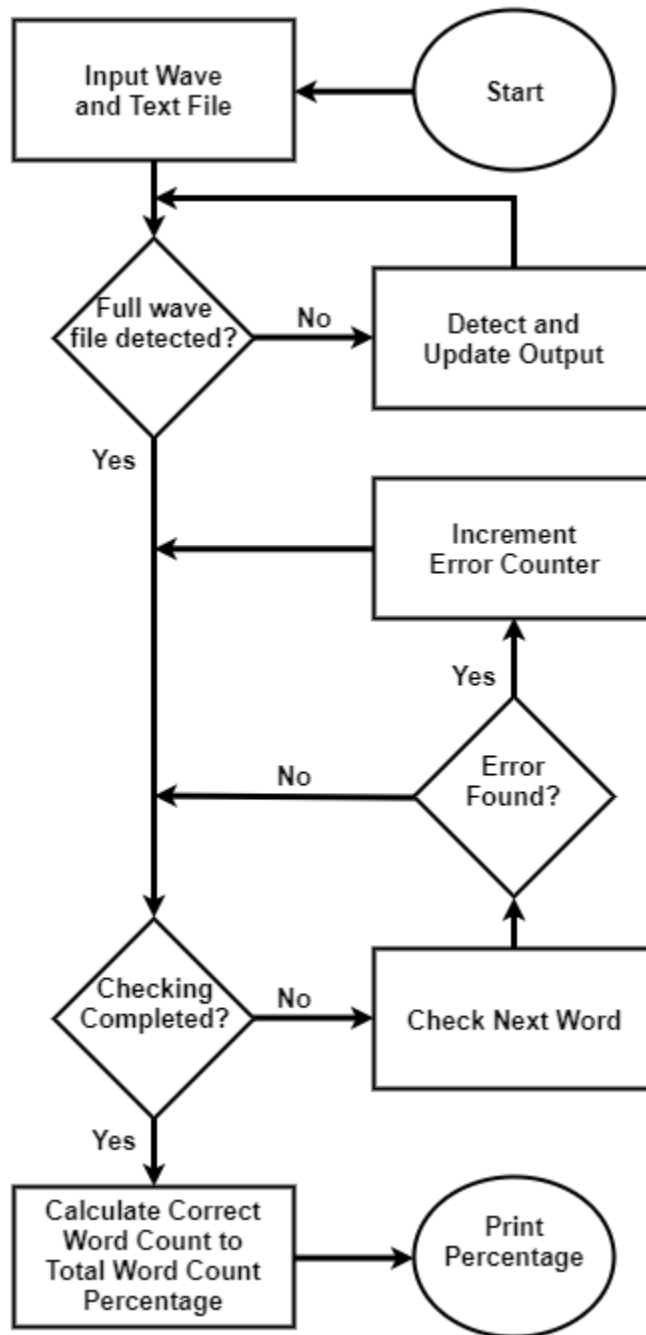


Figure 23. Accuracy Calculation Flowchart

To our knowledge the best work done in this arena was using Microsoft SAPI which obtained a 78% accuracy [4]. But that detection was only on a word by word basis which was a major limitation and the accuracy was for a specific data. Our system not only recognizes the word but also the sentence due to our better training model as well as the newer Sphinx 4 framework.

CHAPTER 05

Conclusion and Future Work

5.1 Conclusion

In this paper we have presented a model for voice to text conversion method for Bengali language using various techniques to refine the data and adapt it for the complexities of implementing Bengali characters. An open source frame work called CMU Sphinx 4 was used to generate Bengali UNICODE font. A digital audio workstation called Audacity was used to manipulate the recorded data. The performance of the proposed model was tested using a dataset where both male and female voices were recorded. The proposed model showed around 75% accuracy for the tested dataset.

5.2 Future Work

If the number of training hours were increased with more speakers then the detection level can be increased dramatically. A good target would be 50 hours of data since the framework recommends that to be the optimum number of hours for a well-trained model. Additionally, this framework supports the Unity game engine in the C# language which opens up many possibilities. Since the Unity engine is able to cross compile C# code to generate native C++ libraries using their own custom IL2CPP technology for over 20 different platforms, it has a much wider support net than Java because it makes the export of native packages very simple [29]. So, if our system were to be deployed through Unity, it could potentially be applicable for almost any platform of our choice with ease, including but not limited to iOS, Android and Windows phone. Custom UI for the interface of usage for our system can easily be built with great flexibility and thanks to the ease of portability, it can be deployed to everyone's hands regardless of which device they have in their pocket.

References

1. Miss. Prachi Khilari, Prof. Bhope V. P., "A Review on Speech to Text Conversion Methods," International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 4 Issue 7, July 2015.
2. B. Raghavendhar Reddy, E. Mahender, "Speech to Text Conversion using Android Platform," International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 Vol. 3, Issue 1, pp.253-258, January -February 2013.
3. Preeti Saini, Parneet Kaur, "Automatic Speech Recognition: A Review," International Journal of Engineering Trends and Technology- Vol 4, Issue 2, 2013.
4. S. Sultana, M. A. H. Akhand, P. K. Das, M. M. H. Rahman, "Bangla Speech-to- Text Conversion using SAPI," In Computer and Communication Engineering (ICCE), International Conference. Kuala Lumpur, 3-5 July 2012. Kuala Lumpur: IEEE. P.385-3, 2012.
5. P. Lamere, P. Kwok, W. Walker, E. Gouva, R. Singh, B. Raj, P. Wolf, "Design of the CMU Sphinx-4 Decoder," Mitsubishi Electric Research Labs, USA, August 2003
6. A. McCallum, "Hidden Markov Models Baum Welch Algorithm," March 9, 2004.
7. M. Pavel, S. Petr, C. Jan, C. Pave, "Phonotactic Language Identification using High Quality Phoneme Recognition," Brno University of Technology, Czech Republic OGI School of Science & Engineering, OHSU, Portland, Oregon USA.
8. N. H. Mohammad, B. Manoj, M. Ghulam, K. Mashud, J. K. Bernd, "Effects of Syllable Language Model on Distinctive Phonetic Features (DPFs) based Phoneme Recognition Performance," International Conference on Computers and Information Technology, 2009. P.285 - 289.
9. K.M. Shivakumar, K. G. Aravind, T. V. Anoop, G. Deepa "Kannada Speech to Text Conversion Using CMU Sphinx," International Conference on Inventive Computation Technologies (ICICT), Vol 3, 2016.
10. Y. E. A. Mohamed, M. M. R. Hafizur, R. W. Mohamed, S. Asadullah, "Building CMU Sphinx language model for the Holy Quran using simplified Arabic Phonemes," Egyptian Informatics Journal, Vol. 17, pp. 305-314, 2016.
11. K. R. Mosur, "Efficient Algorithms for Speech Recognition," University of Milan, May 1996.

12. H. Nazia, S. Dilruba, K. Tarin, "Instant Bangla Speech to Text Conversion," International Journal of Science and Research (IJSR), 2012.
13. M. Sankar, K. D. M. Shyamal, "A BENGALI HMM BASED SPEECH SYNTHESIS SYSTEM," Center for Educational Technology, Indian Institute of Technology Kharagpur, 2014.
14. Md. Abul Hasnat, Jabir Mowla, Mumit Khan, "Isolated and Continuous Bangla Speech Recognition: Implementation, Performance and Application Perspective," Department of Computer Science and Engineering, Conference Papers (Centre For Research on Bangla Language Processing), BRAC University, 2007.
15. J. L. Nusrat, N. E. Qamrun, M. Ghulam, Dr. N. H. Mohammad, Prof. Dr. M. R. Rahman, "Performance Evaluation of Bangla Word Recognition Using Different Acoustic Features," IJCSNS International Journal of Computer Science and Network Security, vol. 10, No. 9, September 2010.
16. Md. R. Mijanur, Md. K. Farukuzzaman, A. M. Mohammad, "Speech Recognition Front-end for Segmentation and Clustering Continuous Bangla Speech," DIU Journal of Science and Technology, vol 5, Issue 1, January 2010.
17. M. Istvan, M. M. Irtraud, "A Linear Memory Algorithm for Baum-Welch Training," BMC Bioinformatics, September 19, 2005.
18. G. Zoubin, I. J. Michael, "Factorial Hidden Markov Models," 1997.
19. P. Kenny, V. Gupta, T. Stafylakis, P. Ouellet and J. Alam, "Deep Neural Networks for extracting Baum-Welch statistics for Speaker Recognition," Centre de Recherche Informatique de Montréal (CRIM), Canada, 2014.
20. Sean R. Eddy, "Multiple alignment using hidden Markov models," Dept. of Genetics, Washington University School of Medicine, 1995.
21. G. F. David Jr, "The Viterbi Algorithm," Proceedings of the IEEE, vol. 61, No. 3, March 1973.
22. W. Adam, "Digital Audio Workstations and Analog to Digital Conversion," Department of Computer Science (CIS) University of Wisconsin-Platteville.
23. N. Francis, "Intonational equivalence: an experimental evaluation of pitch scales," Department of Linguistics, University of Cambridge, UK.

24. A. H. Syed, R. M. Lutfur, A. Farruk, "A Review on Bangla Phoneme Production and Perception for Computational Approaches," 7th WSEAS Int. Conference on Mathematical Methods and Computational Techniques in Electrical Engineering, pp.346-354, October 27-29, 2005.
25. C. Nipa, S. M. Abdus, K. B. Arup, "Separating Words from Continuous Bangla Speech," Global Journal of Computer Science and Technology, p. 172, 2009.
26. M. A. Karim, M. Kaykobad, M. Murshed Monash, "Technical Challenges and Design Issues in Bangla Language Processing," The United States of America by Information Science Reference (an imprint of IGI Global), 2013.
27. B. B. Chaudhuri, U. Pal, "A Complete Printed Bang, A OCR System," Pattern Recognition, Vol. 31, No. 5, pp. 531-549, 1998.
28. "World Population Prospects: The 2017 Revision" United Nations Department of Economic and Social Affairs, Population Division, September, 2017.
29. F. Sebastian, F. Carmen, D. Jozef, "3DRepo4Unity: Dynamic Loading of Version Controlled 3D Assets into the Unity Game Engine," Web3D June, 2017, Brisbane, QLD, Australia.