B.Sc. in Computer Science and Engineering Thesis

Multi Layer Neural Network Based Automatic Word Recognition for Bangla Spoken Language

Submitted by

Md Istiak Ahmed
201114009

Iffatur Ridwan
201114059

Tanvir Ahmed Fuad
201114064

Supervised by

Dr. Mohammad Nurul Huda

Professor & MSCSE Coordinator

Department of Computer Science and Engineering

United International University (UIU)

Department of Computer Science and Engineering
Military Institute of Science and Technology

December 2014

CERTIFICATION

This thesis paper titled **"Multi Layer Neural Network Based Automatic Word Recognition for Bangla Spoken Language"**, submitted by the group as mentioned below has been accepted as satisfactory in partial fulfillment of the requirements for the degree B.Sc. in Computer Science and Engineering in December 2014.

Group Members:

Md Istiak Ahmed

Iffatur Ridwan

Tanvir Ahmed Fuad

Supervisor:

Dr. Mohammad Nurul Huda
Professor & MSCSE Coordinator
Department of Computer Science and Engineering
United International University (UIU)

CANDIDATES' DECLARATION

This is to certify that the work presented in this thesis paper, titled, "Multi Layer Neural Network Based Automatic Word Recognition for Bangla Spoken Language", is the outcome of the investigation and research carried out by the following students under the supervision of Dr. Mohammad Nurul Huda, Professor & MSCSE Coordinator, Department of Computer Science and Engineering, United International University (UIU).

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

Md Istiak Ahmed
201114009

Iffatur Ridwan
201114059

Tanvir Ahmed Fuad
201114064

ACKNOWLEDGEMENT

We are thankful to Almighty Allah for his blessings for the successful completion of our thesis. Our heartiest gratitude, profound indebtedness and deep respect go to our supervisor, Dr. Mohammad Nurul Huda, Professor & MSCSE Coordinator, Department of Computer Science and Engineering, United International University (UIU), for his constant supervision, affectionate guidance and great encouragement and motivation. His keen interest on the topic and valuable advices throughout the study was of great help in completing thesis.

We are especially grateful to the Department of Computer Science and Engineering (CSE) of Military Institute of Science and Technology (MIST) for providing their all out support during the thesis work.

Finally, we would like to thank our families and our course mates for their appreciable assistance, patience and suggestions during the course of our thesis.

Dhaka
December 2014

Md Istiak Ahmed

Iffatur Ridwan

Tanvir Ahmed Fuad

ABSTRACT

Automatic speech recognition (ASR) known as speech recognition is a computer technology that enables a device to recognize and understand spoken words and sentences, by digitizing the sound and matching its pattern against the stored patterns. In short, it is the conversion of spoken speech to text. Currently available devices are largely speaker-dependent and can recognize discrete speech better than the normal (continuous) speech. Speaker independent system recognizes speech of indefinite multiple people. In our research, we have used a system which is speaker independent and can detect continuous speech. Their major applications are in assistive for helping people in working around their disabilities. Our proposed Bangla speech system, based on MFCC+Neural Network+Triphone is a new approach towards the field of Bangla ASR system. For this thesis work, we have prepared a Bangla speech recognition system of Bangla ASR. Most of the Bangla ASR system uses a small number of speakers, but 30 speakers selected from a wide area of Bangladesh, where Bangla is used as a native language, are involved here. In the experiments, Mel-Frequency Cepstral Coefficients (MFCCs) and the result based on (recognized by) Neural Network are inputted to the Hidden Markov Model (HMM) based classifiers for obtaining speech recognition performance. Other than the traditional MFCC triphone model; a new method that have used Neural Network based triphone model had been experimented to get better ASR performance. We used k-mean clustering for the proposed method. From the experimental results, word correct rate and word accuracy for male and female voices distinctly provide much better result for the proposed model based on Neural Network than MFCC-38 as well as MFCC-39. So, our proposed system is in favor of gender independent fact. For male and female voices collectively, sometimes MFCC-39 based model and sometimes Neural Network based model shows better word accuracy and correct rate.

LIST OF ABBREVIATION

AF	Articulatory Features
AM	Acoustic Model
ASR	Automatic Speech Recognition
ATR	Advanced Telecommunication Research Institute International
BPS	Band Pass Filter
DCT	Discrete Cosine Transform
DSR	Distributed Speech Recognition
EM	Expected Maximization
FFT	Fast Fourier Transform
GMM	Gaussian Mixture Model
GI	Gender-Independent
HNN	Hybrid Neural Network
HMM	Hidden Markov Model
HL	Hidden Layer
In/En	Inhibition/Enhancement
BNAS	Bangla Newspaper Article Sentences
LF	Local Feature
LM	Language Model
LR	Linear Regression
MFCC	Mel-Frequency Cepstral Coefficient
NN	Neural Network
OL	Output Layer
OOV	Out-of-Vocabulary
PCR	Phoneme Correct Rate
PRA	Phoneme Recognition Accuracy
SPINE	Speech Recognition In Noisy Environments
SNR	Signal-to-noise Ration

LIST OF SYMBOLS

π	: Initial state distribution
Φ	: Probability distribution
S	: State Sequence

CHAPTER 1
INTRODUCTION

Almost all the major spoken languages in the world have Automatic speech recognition (ASR) systems, but for Bangla (can also be termed as Bengali) too little research has been performed. The lack of proper speech corpus is a major difficulty to research in Bangla ASR. To develop Bangla speech corpus to build a Bangla text to speech system [1] the lack of proper speech corpus is a major issue. However, this effort is a part of developing speech databases for Indian Languages, where Bangla is one of the parts and it is spoken in the eastern area of India (West Bengal and Kolkata as its capital). But in Bangladesh most of the natives of Bangla (more than two thirds) reside, here bangla is the official language. Although both the countries have same written characters of Standard Bangla, there are some sounds that are produced variably in different pronunciations of Standard Bangla. So, there is a need to do research on the main stream of Bangla ASR, which is spoken in Bangladesh. Bangla ASR research or Bangla speech processing can be found in [2–7]. For example, using Hidden Markov Models (HMMs) recognition of isolated and continuous Bangla speech on a small dataset is described in [3]; Bangla vowel characterization is done in [2]; development of Continuous Bangla speech recognition system is in [6], where [7] shows a brief overview of Bangla speech synthesis and recognition. As a whole, most of these works are mainly focused on the on the frequency distributions of different vowels and consonants or simple recognition task on a very small database.

1.1 Contribution

In this work, a medium speech corpus which is based on ASR systems is used for the designing of triphone models. Two stages comprise the method. To catch context of both sides, the first stage designs triphone models; the second stage use Hidden Markov Model based classifier to output word strings based on triphone models. The purpose of this research is to help to build a medium vocabulary triphone based continuous speech recognizer for Bangla language.
In Order to solution some problems in Bangla Speech Recognition, this thesis consentrates on context sensitive triphone model. The problems on which attention is focused are:

a) Co articulation fact,

b) Correction Rate of Word and Sentence,

c) Reduction of mixture component for desired result.

CHAPTER 2
AUTOMATIC SPEECH RECOGNITION

Conversion of human voice into text is the purpose of Automatic Speech Recognition (ASR), which is also termed as Computer Speech Recognition. The task of translating speech is simplified if the voice of the speaker is properly recognized. The recognition system that must be trained to a particular speaker is the main task of most speech recognition software which is referred to the term Voice Recognition.
Again in an expanded sense, the process of enabling a computer to identify and respond to the sounds produced in human speech without being targeted at single speaker such as live TV show on phone request can recognize random voices. This sense can be represented by the term Speech Recognition.
Speech recognition applications include voice user interfaces such as call routing (e.g. I would like to make a collect call), voice dialing (e.g. Call home), domestic appliance control, simple data entry (e.g. entering a credit card number), search (e.g. find a podcast where particular words were spoken), preparation of structured documents (e.g. a radiology report), speech-to-text processing (e.g. word processors or emails), and aircraft (usually termed Direct Voice Input).

2.1 History

While AT&T Bell Laboratories developed a primitive device that could recognize speech in the 1940s, researchers knew that the widespread use of speech recognition would depend on the ability to accurately and consistently perceive subtle and complex verbal input. In 1952 the first speech recognizer is developed and it comprises of a device to recognize single spoken digits [8]. Next in 1964 at New York Worlds Fair, IBM Shoebox another early device was displayed.
Thus, in the 1960s, researchers turned their focus towards a series of smaller goals that would aid in developing the larger speech recognition system. As a first step, developers created a device that would use discrete speech, verbal stimuli punctuated by small pauses. However, in the 1970s, continuous speech recognition, which does not require the user to pause between words, began. This technology became functional during the 1980s and is

still being developed and refined today.
Speech Recognition Systems have become so advanced and mainstream that business and health care professionals are turning to speech recognition solutions for everything from providing telephone support to writing medical reports. Technological advances have made speech recognition software and devices more functional and user friendly, with most contemporary products performing tasks with over 90 percent accuracy.
According to figures provided by industry. Satisfying the needs of consumers and businesses by simplifying customer interaction, increasing efficiency, and reducing operating costs, speech recognition is used in a wide range of applications.
In speech recognition automatic transcription, the constraint which is behind its backwardness is the lacking in the software. The judgment that may be provided by a real human but not yet by an automated system is mostly required as the nature of narrative dictation is highly interpretive. In addition, the requirement of a long period of time to train the software by the user and/or system provider is another visible constraint in this context.
In ASR a comparison is made, to differentiate between artificial syntax systems and natural language processing. The first types of systems stated above are usually domain-specific and the second type of processing stated above are basically language-specific application. Each of these types of application represents its own specific goals and challenges.

2.2 Basics of Speech Recognition

Speech recognition is the way to identify spoken words by using a computer (or other type of machine). In short, it is the interaction between human and computer with the purpose of making it correctly recognizes the words of human voice. A general solution of speech recognition shows in Figure 2.1. Some definitions which are the basics to understand the speech recognition technology are presented below:
Utterance
Vocal expression can be termed as utterance. It is the act or process of producing sounds with the voice that represents a single meaning to the computer. Utterances can be a single word, a few words, a sentence, or even multiple sentences.
Speaker Dependence
It is an Acoustic Model that has been tailored to recognize a particular persons speech. Such Acoustic Models are usually trained using audio from a particular persons speech.
A Speaker Independent Acoustic Model can recognize speech from a person who did not submit any speech audio that was used in the creation of the Acoustic Model.
The reason for the distinction is that it takes much more speech audio training data to create

a Speaker Independent Acoustic Model than a Speaker Dependent Acoustic Model.

Vocabularies

Vocabularies (or dictionaries) are collection of words or utterances to be recognized by the SR system. Usually, larger vocabularies are more difficult to recognize than smaller vocabularies. Here, each entry doesnt have to be a single word. They can be as long as a sentence or two. Smaller vocabularies can have as few as 1 or 2 recognized utterances (e.g. carry on), while very large vocabularies can have a hundred thousand or more.

Accurate

By measuring the accuracy or well recognition utterance, the ability of a recognizer can be examined. This includes not only correct identification of an utterance but also the identification of spoken utterance if it is not in the recognizers vocabulary. Good ASR systems have an accuracy of 98% or more. The acceptable accuracy of a system really depends on the application.

Training

It is the process by which speech recognizer is taught the skills that are needed for the recognition of the speech of a speaker. An ASR system is trained by having the speaker repeat standard or common phrases and adjusting its comparison algorithms to match that particular speaker. Training basically work for the improvement of accuracy of the recognizer.

Training can also be used by speakers that have difficulty speaking, or pronouncing certain words. As long as the speaker can consistently repeat an utterance, ASR systems with training should be able to adapt.

The speech recognition process is represented in Figure 2.2 and will be explained in more

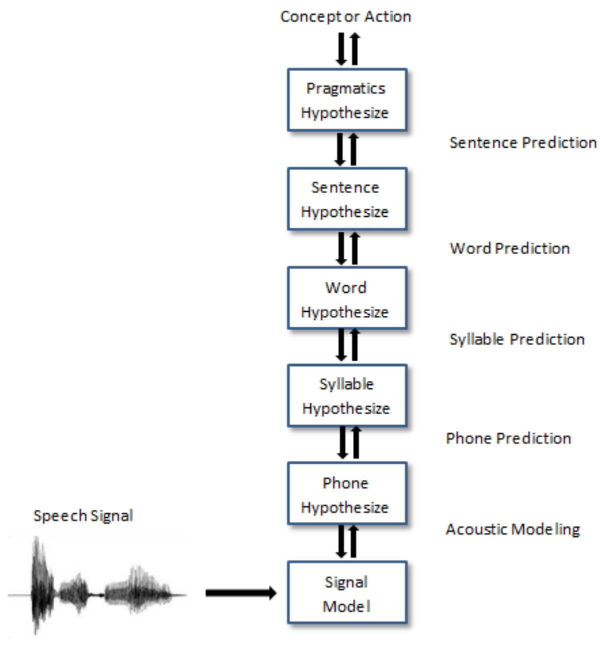


Figure 2.1: General Solution

detail in the following sections.
The speech waveform first undergoes a signal processing step which produces a representation in spectral feature vectors. Phone likelihoods are subsequently estimated, after which a decoding step can finish the recognition process.

2.3 Types of Speech Recognition

With the description of what types of utterances recognizers have the ability to recognize, speech recognition systems can be separated in several different classes. These classes are based on the fact that one of the difficulties of ASR is the ability to determine when a speaker starts and finishes an utterance. Most packages can fit into more than one class, depending on which mode theyre using.
Isolated Words
Isolated word recognizers usually require each utterance to have quiet (lack of an audio signal) on both sides of the sample window. It doesnt mean that it accepts single words, but does require a single utterance at a time. Often, these systems have Listen/Not Listenstates, where they require the speaker to wait between utterances (usually doing processing during

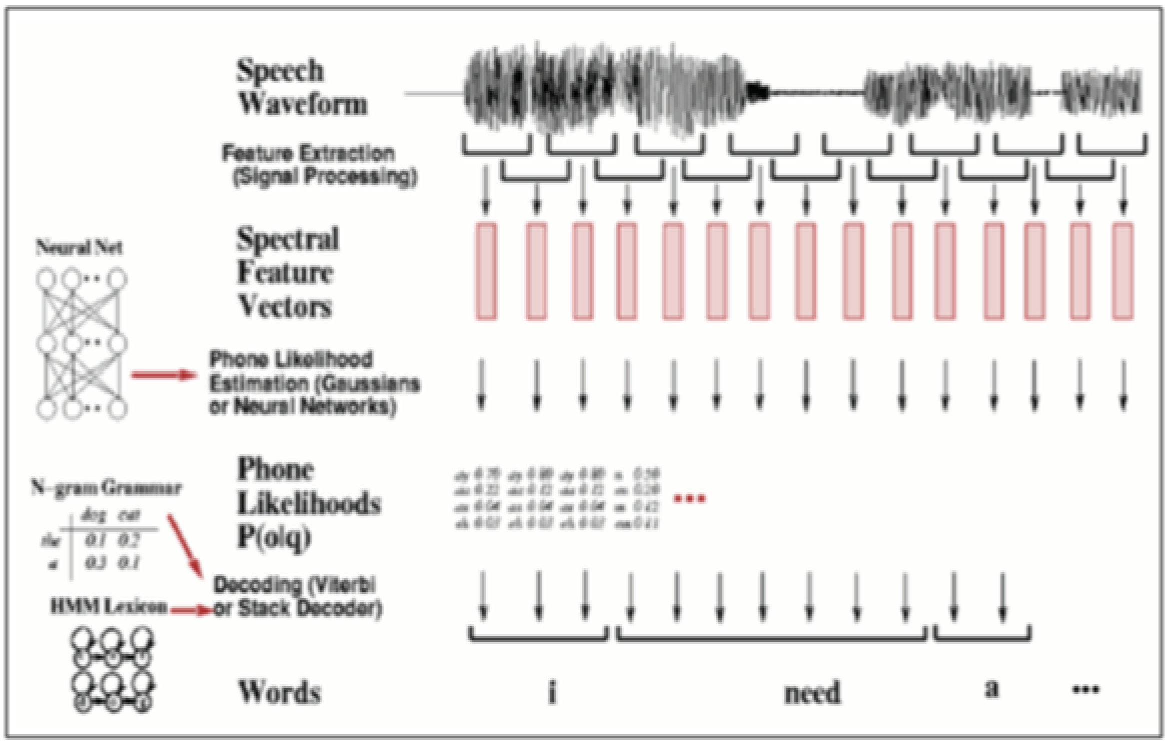

Speech Waveform
Feature Extraction (Signal Processing)
Neural Net
Spectral Feature Vectors
Phone Likelihood Estimation (Gaussians or Neural Networks)
N-gram Grammar
Phone Likelihoods P(o|q)
Decoding (Viterbi or Stack Decoder)
HMM Lexicon
Words
i
need
a
...

Figure 2.2: Schematic architecture for a simplified speech recognizer

the pauses). Isolated Utterance might be a better name for this class.

Connected Words

Connect word systems or connected utterances are similar to Isolated words, but accept separate utterances to be uttered togetherwith a short pause between them.

Continuous Speech

Recognizers which has the capability with continuous speech are some of the most difficult to create because they must utilize special methods to determine utterance boundaries. Continuous speech recognizers allow users to speak almost naturally, while the computer determines the content. It can be suggested as computer dictation.

Spontaneous Speech

There appears to be a variety of definitions for what spontaneous speech actually is. At a basic level, it can be thought of as speech that is natural sounding and not rehearsed. An ASR system with spontaneous speech ability should be able to handle a variety of natural speech features such as words being run together, umsand ahs, and even slight stutters.

The recognition of spontaneous speech can be improved by taking into consideration the effects of the filled pauses while performing the recognition process by:

(1)Either deleting such pauses or by accepting them as words to be added to the dictionary of the ASR system.

(2) Recognizing hesitations and restarts.

(3) By developing the model accuracy at both the acoustic level and at the language model.

(4) Increasing the amount of training data and the lexicon size. This could reduce the error rate without increasing the search complexity.

At the end, for improving the performance of the existing recognizers it is needed to understand the properties of human auditory perception that are relevant for decoding the speech signal and to improve the performance of ASR in different environments is necessary. Also, using longer acoustic units (for example, syllables) instead of using short term speech segments followed by post processing techniques or using dynamic features is promising for the evolution of ASR. Moreover, rich prosodic cues (e.g. fundamental frequency, energy, duration, etc.) that permit successful understanding, which are ignored by state of the art ASR systems, must be considered for better performance. Again, the use of language independent acoustic models and variable ngram language models will enhance the performance further.

CHAPTER 3
BANGLA PHONETIC SCHEME

The International Phonetic Alphabet (IPA) is an alphabetic system of phonetic notation based primarily on the Latin alphabet. It was devised by the International Phonetic Association as a standardized representation of the sounds of spoken language. The IPA is used by foreign language students and teachers, linguists, speech pathologists and therapists, singers, actors, lexicographers, artificial language enthusiasts (conlangers), and translators.
The IPA is designed to represent only those qualities of speech that are distinctive in spoken language: phonemes, intonation, and the separation of words and syllables. To represent additional qualities of speech such as tooth gnashing, lisping, and sounds made with a cleft palate, an extended set of symbols called the Extensions to the IPA may be used.
IPA symbols are composed of one or more elements of two basic types, letters and diacritics. For example, the sound of the English letter t may be transcribed in IPA with a single letter, [t], or with a letter plus diacritics, [th], depending on how precise one wishes to be. Occasionally letters or diacritics are added, removed, or modified by the International Phonetic Association. As of 2008, there are 107 letters, 52 diacritics, and four prosodic marks in the IPA.
In 1886, a group of French and British language teachers, led by the French linguist Paul Passy, formed what would come to be known from 1897 onwards as the International Phonetic Association (in French, l Association phontique internationale). Their original alphabet was based on a spelling reform for English known as the Romic alphabet, but in order to make it usable for other languages, the values of the symbols were allowed to vary from language to language.
Since its creation, the IPA has undergone a number of revisions. After major revisions and expansions in 1900 and 1932, the IPA remained unchanged until the IPA Kiel Convention in 1989. A minor revision took place in 1993, with the addition of four letters for mid-central vowels and the removal of letters for voiceless implosives. The alphabet was last revised in May 2005, with the addition of a letter for a labiodentals flap. Apart from the addition and removal of symbols, changes to the IPA have consisted largely in renaming symbols and categories and in modifying typefaces.
Extensions of the alphabet are relatively recent; Extensions to the IPAwas created in 1990 and officially adopted by the International Clinical Phonetics and Linguistics Association in 1994.

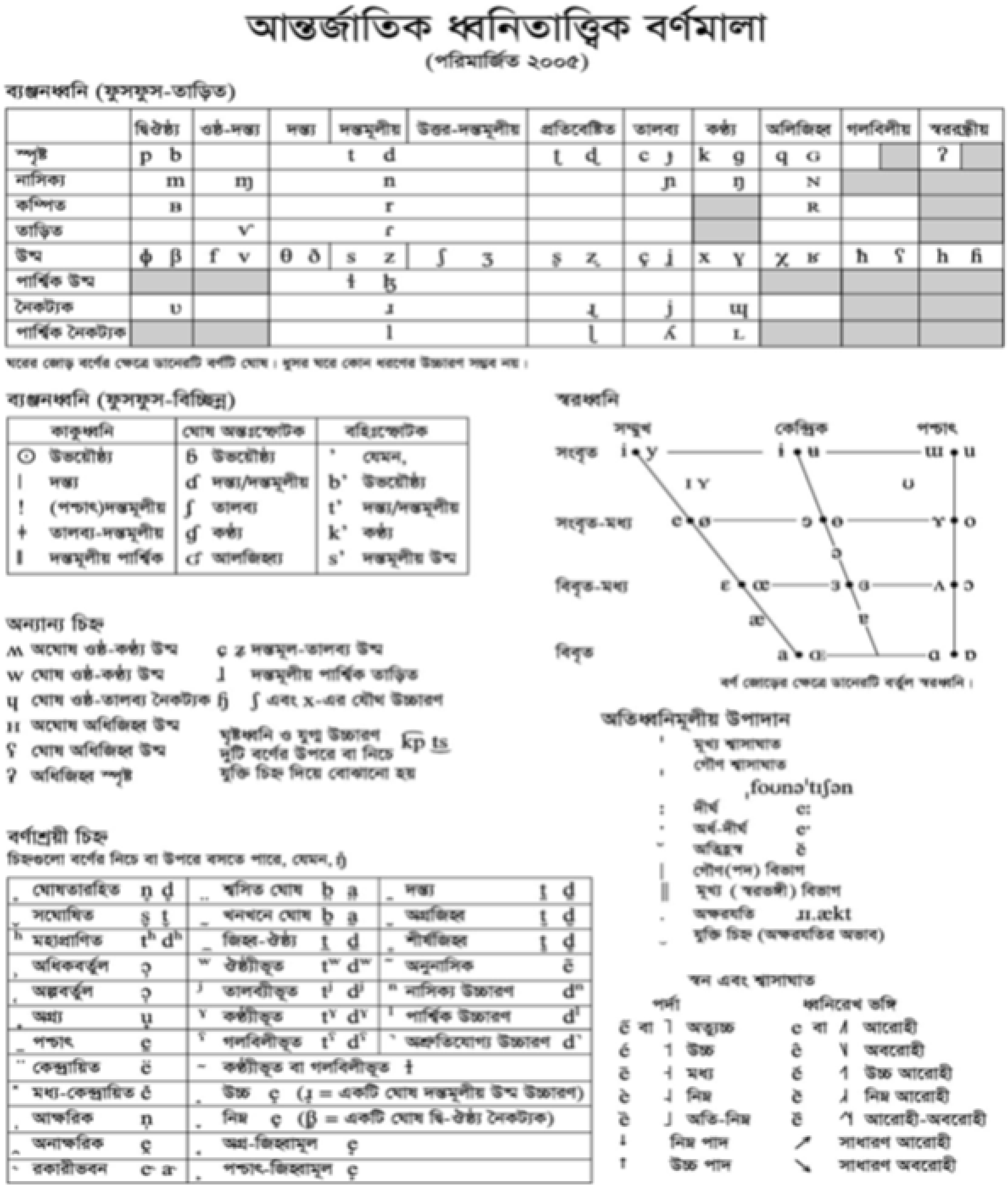
3.1 Bangla Script

The Bengali script (Bengali: bangla lipi) is the writing system for the Bengali language. It is also used, with some modifications, for Assamese, Meitei, Bishnupriya Manipuri, Kokborok, Garo and Mundari languages. All these languages are spoken in the eastern region of South Asia. Historically, the script has also been used to write the Sanskrit language in the same region. From a classificatory point of view, the Bengali script is an abugida, i.e. its vowel graphemes are mainly realized not as independent letters like in a true alphabet, but as diacritics attached to its consonant graphemes. It is written from left to right and lacks distinct letter cases. It is recognizable by a distinctive horizontal line running along the tops of the letters that links them together, a property it shares with two other popular Indian scripts: Devanagari (used for Hindi, Marathi and Nepali) and Gurumukhi (used for Punjabi). The Bengali script is, however, less blocky and presents a more sinuous shaping. The Bengali script evolved from the Eastern Nagari script, which belongs to the Brahmic family of scripts, along with the Devanagari script and other written systems of the Indian subcontinent. Both Eastern Nagari and Devanagari were derived from the ancient Nagari script. In addition to differences in how the letters are pronounced in the different languages, there are some minor typographical differences between the version of the script used for Assamese and Bishnupriya Manipuri as well as Maithili languages, and that used for Bengali and other languages.

The Bengali script was originally not associated with any particular language, but was often used in the eastern regions of Medieval India. It was standardized and modernized by Ishwar Chandra under the reign of the British East India Company. The script was originally used to write Sanskrit, which for centuries was the only written language of the Indian subcontinent in addition to Tamil. Epics of Hindu scripture, including the Mahabharata or Ramayana, were written in older versions of the Bengali script or Mithilakshar/Tirhuta script in this region. After the medieval period, the use of Sanskrit as the sole written language gave way to Pali, and eventually to the vernacular languages we know now as Maithili, Bengali, and Assamese. Srimanta Sankardeva used it in the 15th and 16th centuries to compose his oeuvre in Assamese and Brajavali the language of the Bhakti poets. There is a rich legacy of Indian literature written in this script, which is still occasionally used to write Sanskrit today.

3.2 Bangla IPA Table

The first IPA chart was prepared in 1888 by the earliest form of the International Phonetic Association and it has gone through many changes since then. The 1888 chart was rather a list of symbols and their descriptions.
The latest, revised 2005, in Bengali shown in figure 3.1.



আন্তর্জাতিক ধ্বনিতাত্ত্বিক বর্ণমালা
(পরিমার্জিত ২০০৫)

ব্যঞ্জনধ্বনি (ফুসফুস-তাড়িত)

	দ্বিঔষ্ঠ্য	ওষ্ঠ-দন্ত্য	দন্ত্য	দন্তমূলীয়	উত্তর-দন্তমূলীয়	প্রতিবেষ্টিত	তালব্য	কণ্ঠ্য	অলিজিহ্ব	গলবিলীয়	স্বরতন্ত্রীয়
স্পৃষ্ট	p b			t d		ʈ ɖ	c ɟ	k g	q ɢ	▨	ʔ ▨
নাসিক্য	m	ɱ		n		ɳ	ɲ	ŋ	ɴ	▨	▨
কম্পিত	ʙ			r				▨	ʀ		▨
তাড়িত		ⱱ		ɾ		ɽ		▨			▨
উষ্ম	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ʝ	x ɣ	χ ʁ	ħ ʕ	h ɦ
পার্শ্বিক উষ্ম	▨	▨		ɬ ɮ					▨	▨	▨
নৈকট্যক	ʋ			ɹ		ɻ	j	ɰ		▨	▨
পার্শ্বিক নৈকট্যক	▨	▨		l		ɭ	ʎ	ʟ	▨	▨	▨

স্বরের জোড় বর্ণের ক্ষেত্রে ডানেরটি ঘোষ। ধূসর ঘরে কোন ধরনের উচ্চারণ সম্ভব নয়।

ব্যঞ্জনধ্বনি (ফুসফুস-বিচ্ছিন্ন)

কাকুধ্বনি	ঘোষ অন্তঃস্ফোটক	বহিঃস্ফোটক
ʘ উভয়ৌষ্ঠ্য	ɓ উভয়ৌষ্ঠ্য	ʼ যেমন,
ǀ দন্ত্য	ɗ দন্ত্য/দন্তমূলীয়	pʼ উভয়ৌষ্ঠ্য
ǃ (পশ্চাৎ)দন্তমূলীয়	ʄ তালব্য	tʼ দন্ত্য/দন্তমূলীয়
ǂ তালব্য-দন্তমূলীয়	ɠ কণ্ঠ্য	kʼ কণ্ঠ্য
ǁ দন্তমূলীয় পার্শ্বিক	ʛ অলিজিহ্ব্য	sʼ দন্তমূলীয় উষ্ম

স্বরধ্বনি

বর্ণ জোড়ের ক্ষেত্রে ডানেরটি বর্তুল স্বরধ্বনি।

অন্যান্য চিহ্ন

ʍ অঘোষ ওষ্ঠ-কণ্ঠ্য উষ্ম	ɕ ʑ দন্তমূল-তালব্য উষ্ম
w ঘোষ ওষ্ঠ-কণ্ঠ্য উষ্ম	ɺ দন্তমূলীয় পার্শ্বিক তাড়িত
ɥ ঘোষ ওষ্ঠ-তালব্য নৈকট্যক	ɧ ʃ এবং x-এর যৌথ উচ্চারণ
ʜ অঘোষ অধিজিহ্ব উষ্ম	যুক্তধ্বনি ও যুগ্ম উচ্চারণ দুটি বর্ণের উপরে বা নিচে যুক্তি চিহ্ন দিয়ে বোঝানো হয় k͡p t͜s
ʢ ঘোষ অধিজিহ্ব উষ্ম	
ʡ অধিজিহ্ব স্পৃষ্ট	

অতিধ্বনিমূলীয় উপাদান

ˈ	মুখ্য শ্বাসাঘাত	
ˌ	গৌণ শ্বাসাঘাত	ˌfoʊnəˈtɪʃən
ː	দীর্ঘ	eː
ˑ	অর্ধ-দীর্ঘ	eˑ
˘	অতিহ্রস্ব	ĕ
|	গৌণ(পদ) বিভাগ	
‖	মুখ্য (স্বরভঙ্গী) বিভাগ	
.	অক্ষরযতি	ɹi.ækt
‿	যুক্তি চিহ্ন (অক্ষরযতির অভাব)	

বর্ণাশ্রয়ী চিহ্ন
চিহ্নগুলো বর্ণের নিচে বা উপরে বসতে পারে, যেমন, ŋ̊

̥ ঘোষতারহিত n̥ d̥	̤ শ্বসিত ঘোষ b̤ a̤	̪ দন্ত্য t̪ d̪
̬ সঘোষিত s̬ t̬	̰ খনখনে ঘোষ b̰ a̰	̺ অগ্রজিহ্ব t̺ d̺
ʰ মহাপ্রাণিত tʰ dʰ	̼ জিহ্ব-ঔষ্ঠ্য t̼ d̼	̻ শীর্ষজিহ্ব t̻ d̻
̹ অধিকবর্তুল ɔ̹	ʷ ঔষ্ঠীকৃত tʷ dʷ	̃ অনুনাসিক ẽ
̜ অল্পবর্তুল ɔ̜	ʲ তালব্যীকৃত tʲ dʲ	ⁿ নাসিক্য উচ্চারণ dⁿ
̟ অগ্র u̟	ˠ কণ্ঠীকৃত tˠ dˠ	ˡ পার্শ্বিক উচ্চারণ dˡ
̠ পশ্চাৎ e̠	ˤ গলবিলীকৃত tˤ dˤ	̚ অশ্রুতিযোগ্য উচ্চারণ d̚
̈ কেন্দ্রায়িত ë	̴ কণ্ঠীকৃত বা গলবিলীকৃত ɫ	
̽ মধ্য-কেন্দ্রায়িত e̽	̝ উচ্চ e̝ (ɹ̝ = একটি ঘোষ দন্তমূলীয় উষ্ম উচ্চারণ)	
̩ আক্ষরিক n̩	̞ নিম্ন e̞ (β̞ = একটি ঘোষ দ্বি-ঔষ্ঠ্য নৈকট্যক)	
̯ অনাক্ষরিক e̯	̘ অগ্র-জিহ্বামূল e̘	
˞ রকারীভবন ɚ a˞	̙ পশ্চাৎ-জিহ্বামূল e̙	

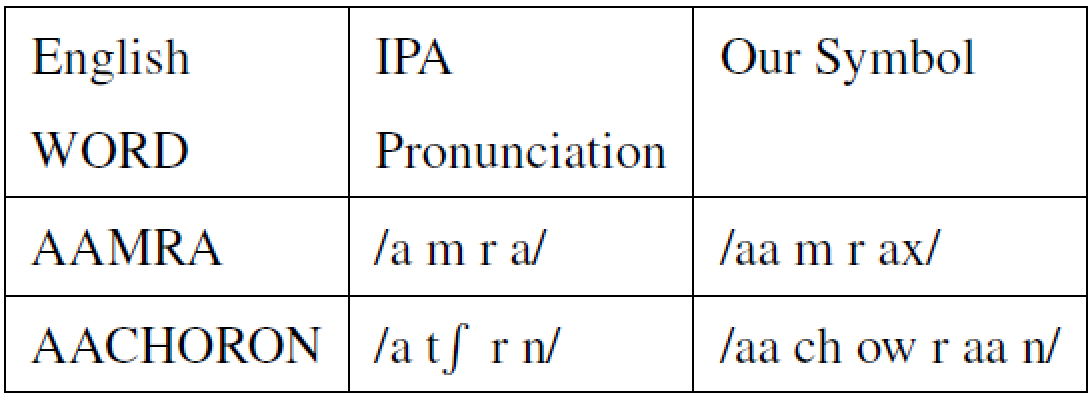
স্বন এবং শ্বাসাঘাত

পর্দা		ধ্বনিরেখ ভঙ্গি	
e̋ বা ˥	অত্যুচ্চ	ě বা ˩˥	আরোহী
é ˦	উচ্চ	ê ˥˩	অবরোহী
ē ˧	মধ্য	e᷄ ˦˥	উচ্চ আরোহী
è ˨	নিম্ন	e᷅ ˩˨	নিম্ন আরোহী
ȅ ˩	অতি-নিম্ন	e᷈ ˧˦˧	আরোহী-অবরোহী
ꜜ	নিম্ন পাদ	↗	সাধারণ আরোহী
ꜛ	উচ্চ পাদ	↘	সাধারণ অবরোহী

Figure 3.1: A chart of the full International Phonetic Alphabet

3.3 Bangla Phoneme Schemes

The Phonetic inventory of Bangla consists of 14 vowels, including seven nasalized vowels, and 29 consonants. Native Bangla words do not allow initial consonant clusters: the maximum syllable structure is CVC (i.e. one vowel flanked by a consonant on each side). Sanskrit words borrowed into Bangla possess a wide range of clusters, expanding the maximum syllable structure to CCCVC. English or other foreign borrowings add even more cluster types into the Bangla inventory.
In the Bengali script, clusters of consonants are represented by different and sometimes quite irregular characters; thus, learning to read the script is complicated by the sheer size of the full set of characters and character combinations, numbering about 350. While efforts at standardizing the script for the Bengali language continue in such notable centers as the Bangla Academies (unaffiliated) at Dhaka (Bangladesh) and Kolkata (West Bengal, India), it is still not quite uniform as yet, as many people continue to use various archaic forms of letters, resulting in concurrent forms for the same sounds. Among the various regional variations within this script, only the Assamese and Bengali variations exist today in the formalized system. It seems likely that the standardization of the script will be greatly influenced by the need to typeset it on computers. The large alphabet can be represented, with a great deal of ingenuity, within the ASCII character set, omitting certain irregular conjuncts. Work has been underway since around 2001 to develop Unicode fonts, and it seems likely that it will split into two variants, traditional and modern.
In this and other articles on Wikipedia dealing with the Bengali language, a Romanization scheme used by linguists specializing in Bengali phonology is included along with IPA transcription. A recent effort by the government of West Bengal focused on simplifying Bengali spellings in primary school texts.



English WORD	IPA Pronunciation	Our Symbol
AAMRA	/a m r a/	/aa m r ax/
AACHORON	/a tʃ r n/	/aa ch ow r aa n/

Figure 3.2: Some Bangla words with their orthographic transcriptions and IPA

CHAPTER 4
HMM-BASED CLASSIFIER

4.1 About HMM

HMM phoneme models basically consists of three emitting states and a simple left-to-right topology as illustrated in Figure 4.1 and Figure 4.2. To ease joining of the models together the entry and exit states are provided. The exit state of one phoneme model can be connected with the entry state of another to from a composite HMM. This allows phone models to be joined together to form words and words to be joined together to cover complete utterances.

4.2 Modeling of HMM

Generally, an HMM is specified by a five-tuple:
(S, O, π, A, B)
(1) $S = \{1, 2, ..., ; N\}$, set of hidden states
N: the number of states.

S_t : the state at time t.

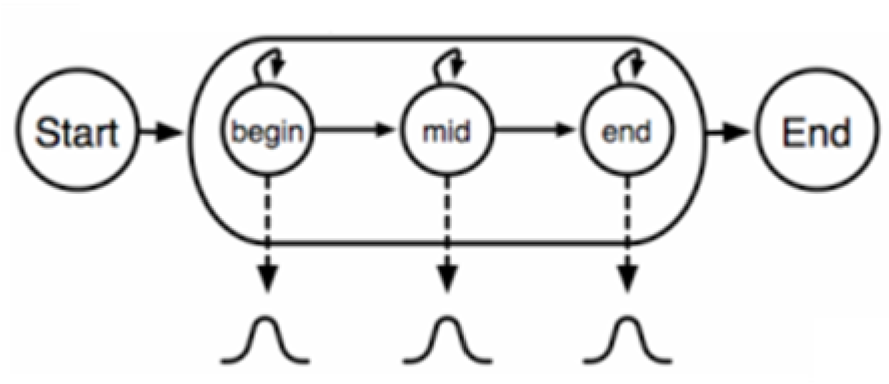


Figure 4.1: Standard HMM phoneme model (without Gaussian mixtures)

(2) $O = \{o_1, o_2, ..., o_M\}$, set of observation symbols
M: the number of observation symbols
(3) $\pi = \{\pi_i\}\pi_i = P(s_0 = i)1 \leq i \leq N$, the initial state distribution
(4) $A = \{a_ij\}a_ij = P(s_t = j \mid s_{t-1} = i), 1 \leq i, j \leq N$
State transition probability distribution.
(5) $B = \{b_j(K)\}b_j(k) = p(X_t = o_k \mid s_t = j)1 \leq j \leq N, 1 \leq k \leq M$ Observation symbol probability distribution in state.
To sum up, a complete specification of an HMM includes:
(i) two constant-size parameters: N and M (representing the total number of states and the size of observation symbols)
(ii) three sets of probability distribution: $\Phi = (A, B, \pi)$

4.3 Three basic problems of HMM

HMM has an issue of facing three major problem. They are :
1. The evaluation problem : deals with the probability of the model.
2. The decoding problem : deals with the most likely state sequence.
3. The learning problem : deals with the adjustment of the model parameter to maximize the joint probability.
A brief definition of these problems are given below :
(i) The evaluation problem:
Given a model Φ and a sequence of observation $X = (X_1, X_2, ..., X_T)$, what is the probability $P(X \mid \Phi)$; i.e, the probability of the model that generates the observations?
(ii) The decoding problem:
Given a model Φ and a sequence of observation $X = (X_1\, X_2 \,....\, X_T)$, what is the most

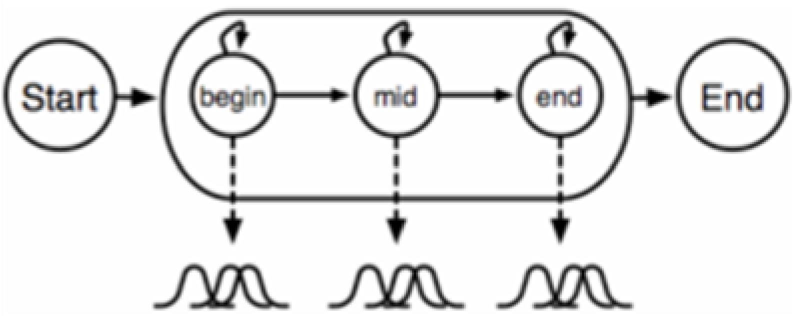


Figure 4.2: Standard HMM phoneme model (using Gaussian mixtures)

likely state sequence $S = (s_0, s_1, ..., s_T)$ in the model that produces the observations?
(iii) The learning problem:
Given a model Φ and a set of observations, how can we adjust the model parameter to maximize the joint probability $\prod P(X \mid \Phi)$?

4.4 Solutions of the problems in HMM

As solutions to each of the three problems in HMM, three basic algorithms are used. These algorithms are :
1. Forward algorithm : define forward probability
2. Vitebri algorithm : define the best path
3. Baum-Welch algorithm : iteratively recomputes the model parameters to increase the likelihood of the training data at each iteration.
The above algorithms stated above are described below :
(i) First problem solution [Forward algorithm]
Define forward probability, $\alpha_t = P(X_1^t, s_t = i \mid \Phi)$
$\alpha_t(i)$ is the probability that the HMM is in state i having generated partial observation $X_1^t (namely X_1, X_2, ..., X_t$
(ii) Second problem solution: [Viterbi Algorithm]

Instead of summing up probabilities from different paths coming to the same destination state, the Viterbi algorithm picks and remembers the best path.
Define the best path probability,

$$V_t(i) = P(X_1^t, S_1^{t-1}, s_t = i \mid \Phi)$$

$V_t(i)$ is the probability of the most likely state sequence at time t, which generates the observation $X_1^t (untiltimet)$ and ends in state i. (iii) Third problem solution: [Baum-Welch Algorithm]
Baum-Welch Algorithm or so-called Forward-Backward algorithm in essence is an EM (Expectation Maximization) algorithm. The basic idea of EM algorithm is to iteratively recomputed the model parameters given their current estimates so as to increase the likelihood of the training data at each iteration.

CHAPTER 5
CONTEXT DEPENDENT TRIPHONE MODEL

Context-independent models or monophone models, assume that a phoneme in any context is equivalent to the same phoneme in any other context. Since the articulators do not move from one position to another immediately in most phoneme transitions this assumption is fundamentally proved incorrect. The transition duration is variable and depends on the phonemes, For example, for /v/ and /j/ the transitions are very long but from stop consonants to vowels the transition is significantly shorter but by no means discrete. Thus, neighboring phonemes are bound to have an effect on the examined phoneme. So, it should be considered that these co-articulation effects caused by context-dependency should be taken into account.
The terms precontext and postcontext are used to indicate the preceding and following neighbor of a phoneme. Figure 5.1 shows these terms for the triphone ch-aa+d.
Using word models is one way of dealing with the dependency. They are suitable for small vocabulary task and have been shown to be more accurate than phoneme models [9]. But for large recognizer they are not feasible. Instead, subword models are likely to produce better results.
In the past, a natural shift occurred from word models to syllable models. They were proposed already in the mid 70s by Fujimura [10]. The early syllable recognizers were non-HMM based. Now, demi-syllable and biphone models (a unit consisting of two consecutive phonemes) have been used. Since phonemes are abstract super classes of phones, classification can be performed with different precision. As a result, some variations in the number of phonemes are caused. Syllable models have been popular especially in Mandarin, Can-

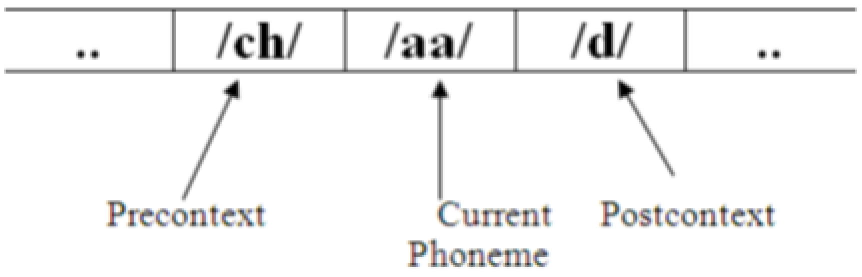


Figure 5.1: The definition of pre- and postcontext for the triphone ch-aa+d

tonese, and Japanese recognizers but they are suitable for other languages as well. Diphones (phoneme units dependent on either pre- or postcontext) were introduced in the late 1970s and early 1980s [11,12]. A few years after that, Schwartz et al. proposed the use of triphones in speech recognition [10]. This work concentrates on modeling triphones. This selection was made based on the fact that the most important coarticulatory effects on a phoneme are due to its immediate neighbors on either side. Moreover, triphones are commonly used in large vocabulary recognition in other languages, but not so much in Finnish.

5.1 Fundamental Mechanism of Triphones

To model the co-articulatory effects between phonemes and that there was nothing special about the units themselves, Schwartz [10] noted that all the subword units longer than phonemes (biphones, syllables, etc.) applied as units to speech recognition were merely trying. This motivated him to return to modeling phonemes. Only this time they were made context dependent, which led to the introduction of the concept of triphones - a model for a single phoneme conditioned on its preceding and following neighbor phoneme. The idea of triphones is used in any modern recognizer, and in [13] the actual results of the first recognizer utilizing triphones are presented. A triphone is simply a model of a single phoneme conditioned on its immediate neighbors, and not a structure of three phonemes. Similarly, a diphone is a model of a phoneme conditioned on either its left or right phoneme or a quinphone is conditioned on two neighboring phonemes on either side.

Context-dependent models can be constructed in two ways: they can either be word internal or cross-word. For cross-word triphones the phonemes at the end or beginning of neighboring words are considered to affect the phoneme. On the contrary, when constructing word-internal models, context beyond the word borders are not considered. Usually, the number of cross-word triphones is considerably higher than the number of word-internal triphones.

The cross-word triphones are a natural choice for continuous speech recognition, since there are seldomly clear pauses between words in fluent speech. Actually, a stop consonant might introduce a longer pause than a word break. The problem, again, is the increasing number of models and the shortage of data for training them.

5.2 For Context-dependent HMMs Clustering Mechanisms

In two cases a set of full triphones is immoderate. The first case arise in all practical cases there is not enough training material for many of the triphones. The second case is, despite co-articulatory effects some triphones are quite similar and had better be covered by the same model. With the minimization of the number of parameters the training data problem can be solved. It can be performed in several ways. The number of models or the number of states can be reduced by state or model clustering. Another approach is confining parameters inside the states or models, that is, forcing them to be equal for two different states (means and variances) or models (transition matrices). A straightforward way of reducing the number of parameters in a triphone model set could be to tie all the parameters of all models center states. The assumption that the center of each triphone (for the same phoneme) is similar could lead to this kind of an approach. Clustering mechanisms provide better results than this kind of direct tying. Some clustering algorithms are depicted in this section.

5.2.1 Data-driven (bottom-up) Clustering

In the data driven clustering algorithm each state is initially located in its respective cluster. Next, two clusters which form the smallest cluster are amalgamated together. This amalgamation of clusters is iteratively continued until the smallest cluster that would be constructed by combining any two clusters would be greater than some predefined limit.
The size of the cluster is represented as the longest distance between any two members of the cluster. The metric is termed as the Euclidean distance between the means of the two states. A constraint in this fact is its limitation to deal with invisible triphones (triphones not present in the training data), which are bound to occur in large vocabulary recognition with cross-word triphones. Therefore, diphone and monophone models are normally used to deal with this problem.
This algorithm is bottom-up since it starts with individual states and ends with clusters.

An illustration of the algorithm is depicted in Figure 5.2. This clustering algorithm was introduced in [14].

5.2.2 Decision-tree (top-down) Clustering

Binary decision trees [15] is another perspective for clustering states. Furthermore, to states and unlike data-driven clustering described above, this algorithm can be used to cluster

entire models as well.
To split the clusters during the process a set of questions regarding phonemes context is needed. There is no specification regarding the number of questions. As an example a typical question might be: Is the left context of this phoneme either an / a/ or an /o/?.
A brief description of the algorithm stated below: initially in the root node of a tree, all states/models in a given list are placed. The nodes are iteratively split by selecting a question. Based on the answer, states/models in the state are placed either in the right or left child node of the current node. It is performed iteratively until the log likelihood increase of the states/models in the tree node obtained by the best question is below a predefined limit. At this stage, all the parameters in the state/model are tied. An illustration is in Figure 5.3.

The question used is chosen to maximize the likelihood of the training data given the final set of model/state tying. When the node is split, the likelihood of its child nodes is bound to increase since the number of parameters to describe the same data increases. The log likelihood can be calculated based on the statistics (means, variances, and state occupation counts) gathered from the training data, and based on that information the best question for each node can be chosen.

5.2.3 Classification Based on Articulatory Facts

For the classification of triphone models one of the criterion is, to use decisions made a priori about the context for classifications. Basically, one decides classifications for phonemes and then classifies those triphones with contexts from the same phoneme classes to belong to

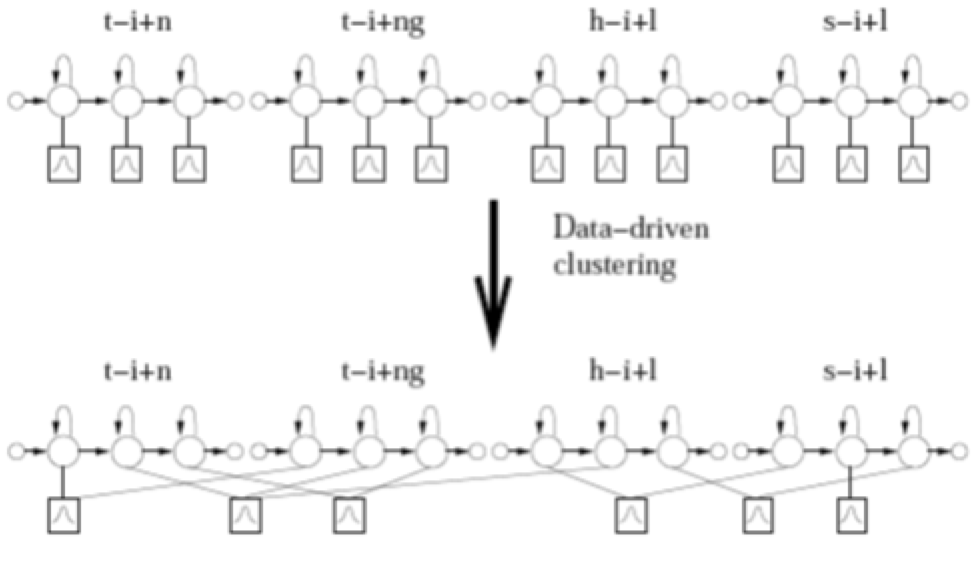


Figure 5.2: Data-driven state clustering for some triphones of the phoneme /i/

the same broad class triphone (or cluster). The phoneme classes could be formed randomly but spontaneously it would be beneficial if there was some similarity between the members of a phoneme class. Furthermore, natural choices are based on articulatory facts. This type of approach has been suggested in [16] and [17]. Here, two different classifications were used: one is based on the type of the phoneme (short: ToP) and the other on the place of articulation (short: PoA). articulation (short: PoA).

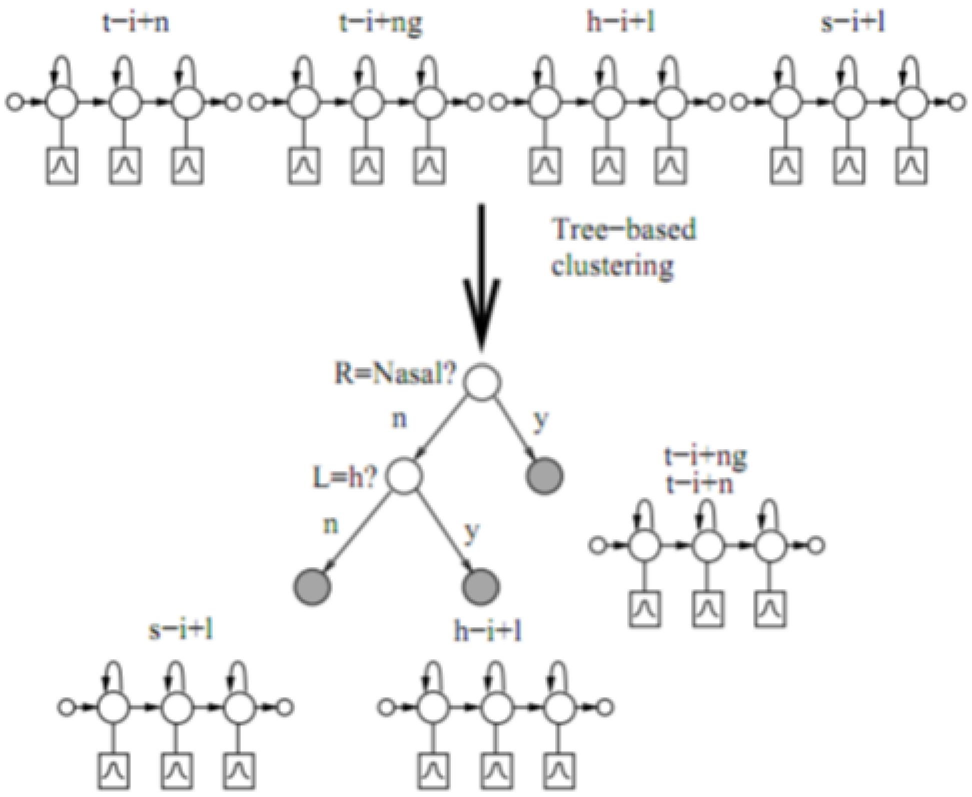


Figure 5.3: Part of the tree-based model clustering process of /i/-triphones. Leaf nodes are gray, and they form the final clusters

CHAPTER 6
PROPOSED METHOD

Automatic speech recognition (ASR) deals with the decoding of an acoustic signal of a speech utterance into corresponding text transcription, such as words, phonemes or other language units. Even after years of extensive research and development, accuracy in ASR remains a challenge to researchers. There are number of well known factors which determine accuracy. The prominent factors are those that include variations in context, speakers and noise in the environment. Therefore, research in ASR has many open issues with respect to small or large vocabulary, isolated or continuous speech, speaker dependent or independent and environmental robustness.
ASR for western languages like English and Asian languages like Chinese are well matured. But similar research in Bangla (widely used as Bengali) languages is still in its infancy stage. Another major hurdle in ASR for the Bangla language is resource deficiency. Annotated speech corpora for training and testing the acoustic models are scarce. Recently there is a growing interest in ASR for Bangla language [2–7]. Continuous Bangla speech recognition system is developed in [5], while [18] presents a brief overview of Bangla speech synthesis and recognition. However, most of these researches have some problems:
(i) deals with small scale speech corpus,
(ii) use only time domain information and
(iii) constructs triphone models [18–22] using MFCC features and consequently, better recognition performance is not obtained.
In this study, we have proposed an ASR system where information is based on time domain and frequency domain. The proposed method comprises three stages, where the first stage is to extract the acoustic features for the given speech example. The second stage is to train and test the neural network by using the acoustic features extracted from the previous stage. Third and final stage is to use the HTK (Hidden Markov Model Toolkit) with the output form the neural network.

The steps we have followed in this case is explained below:

1. Make a folder Experiment (any name can be used) into the drive D (any drive can be used but this drive name must use in the command).

2. Make ensure the Train.scp (this has to be made according to the input voice and any name can be used instead of Train but extension should be .scp), config.txt, HCopy.exe files are into the folder Experiment.

3. In Train.scp (in this experiment we used 3000 input speech which contains in 30 different folders and each folder has got 100 inputs voice) the information has to typed look like the following. Here left part is the input and right part is the output in figure 6.1.
* different folder name can be used for inputs and outputs. But it should be write correctly in Train.scp

4. Now run the following command.
$HCopy - T1 - Cconfig.txt - STrain.scp$

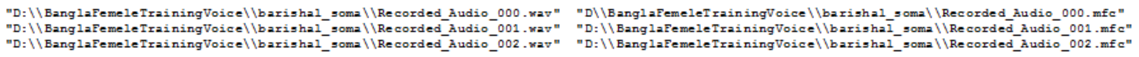
5. Now make source.mfc (any name can be used instead of source but extension should be .mfc) into the folder practice and type the following information
$D : \backslash\backslash Experiment \backslash\backslash BanglaFemeleTrainingVoice \backslash\backslash barishal\_soma$
$\backslash\backslash Recorded\_Audio\_000.mfc$
$D : \backslash\backslash Experiment \backslash\backslash BanglaFemeleTrainingVoice \backslash\backslash barishal\_soma$
$\backslash\backslash Recorded\_Audio\_001.mfc$
$D : \backslash\backslash Experiment \backslash\backslash BanglaFemeleTrainingVoice \backslash\backslash barishal\_soma$
$\backslash\backslash Recorded\_Audio\_002.mfc$
* Here it is noted that the above information are taken from the right part of the step 3 but here will not be used.

6. Put HList.exe into the folder Experiment.

7. Put Batch_HList.m into the folder Experiment and open this file using matlab.

8. After executing Batch_HList.m, the following file will be generated

a. dest.mfc into the folder Experiment which contain the following information



```
"D:\\BanglaFemeleTrainingVoice\\barishal_soma\\Recorded_Audio_000.wav"  "D\\BanglaFemeleTrainingVoice\\barishal_soma\\Recorded_Audio_000.mfc"
"D:\\BanglaFemeleTrainingVoice\\barishal_soma\\Recorded_Audio_001.wav"  "D:\\BanglaFemeleTrainingVoice\\barishal_soma\\Recorded_Audio_001.mfc"
"D:\\BanglaFemeleTrainingVoice\\barishal_soma\\Recorded_Audio_002.wav"  "D:\\BanglaFemeleTrainingVoice\\barishal_soma\\Recorded_Audio_002.mfc"
```

Figure 6.1: Train.scp Example

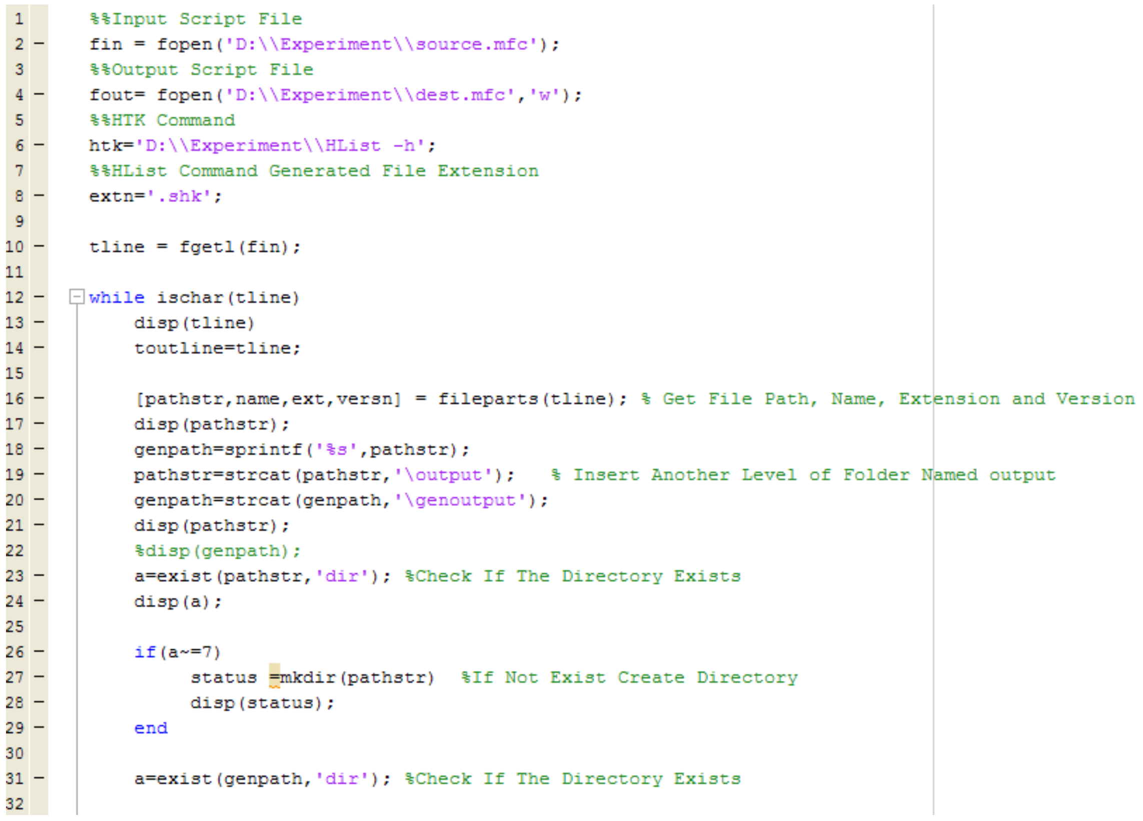
$D:\backslash\backslash Experiment\backslash\backslash BanglaFemeleTrainingVoice\backslash\backslash barishal\_soma$
$\backslash\backslash output\backslash\backslash Recorded\_Audio\_000.shk$
$D:\backslash\backslash Experiment\backslash\backslash BanglaFemeleTrainingVoice\backslash\backslash barishal\_soma$
$\backslash\backslash output\backslash\backslash Recorded\_Audio\_001.shk$
$D:\backslash\backslash Experiment\backslash\backslash BanglaFemeleTrainingVoice\backslash\backslash barishal\_soma$
$\backslash\backslash output\backslash\backslash Recorded\_Audio\_002.shk$

b. output and genoutput folder will be created into

$D:\backslash Experiment\backslash BanglaFemeleTrainingVoice\backslash barishal\_soma$
$\backslash andinoutputfolderwewillgetautomaticallythefollowingfiles:$
$Recorded\_Audio\_000.shk, Recorded\_Audio\_001.shk,$

* Here it is noted that

$D:\backslash Experiment\backslash BanglaFemeleTrainingVoice\backslash barishal\_soma$
$\backslash genoutput$
will not contain any information at that time.



```
%%Input Script File
fin = fopen('D:\\Experiment\\source.mfc');
%%Output Script File
fout= fopen('D:\\Experiment\\dest.mfc','w');
%%HTK Command
htk='D:\\Experiment\\HList -h';
%%HList Command Generated File Extension
extn='.shk';

tline = fgetl(fin);

while ischar(tline)
    disp(tline)
    toutline=tline;

    [pathstr,name,ext,versn] = fileparts(tline); % Get File Path, Name, Extension and Version
    disp(pathstr);
    genpath=sprintf('%s',pathstr);
    pathstr=strcat(pathstr,'\output');   % Insert Another Level of Folder Named output
    genpath=strcat(genpath,'\genoutput');
    disp(pathstr);
    %disp(genpath);
    a=exist(pathstr,'dir'); %Check If The Directory Exists
    disp(a);

    if(a~=7)
        status =mkdir(pathstr)   %If Not Exist Create Directory
        disp(status);
    end

    a=exist(genpath,'dir'); %Check If The Directory Exists

```

Figure 6.2: Batch_HList.m

9. Put line_cutter.cpp file into the folder Line-Cutter (any name can be used) which have to be created into the folder Experiment.

10. Put dest.mfc (which we created in the last step#8) into the folder Line-Cutter.

11. Run line_cutter.cpp. after executing this program the following information will be generated

a. Out_vec.test into the folder Line-Cutter. Out_vec.test (open it and remove last blank line and save it) file contain the following information:

$D : \backslash\backslash Experiment\backslash\backslash BanglaFemeleTrainingVoice\backslash\backslash barishal\_soma$
$\backslash\backslash genoutput\backslash\backslash Recorded\_Audio\_000.gen$
$D : \backslash\backslash Experiment\backslash\backslash BanglaFemeleTrainingVoice\backslash\backslash barishal\_soma$
$\backslash\backslash genoutput\backslash\backslash Recorded\_Audio\_001.gen$
$D : \backslash\backslash Experiment\backslash\backslash BanglaFemeleTrainingVoice\backslash\backslash barishal\_soma$
$\backslash\backslash genoutput\backslash\backslash Recorded\_Audio\_002.gen$

b. In

$D : \backslash Experiment\backslash BanglaFemeleTrainingVoice\backslash barishal\_soma$
$\backslash genoutputfolder$
we will get automatically the following files:

$Recorded\_Audio\_000.gen, Recorded\_Audio\_001.gen,$

12. Make a folder Neural_Network. Put label.txt & phoneme.txt in this folder. Put dimension_117_generator.cpp in this folder. Also put Out_vec.test in this folder which was created in the last step. Now open and run dimension_117_generator.cpp. This program will execute the following file-
a. Input.scp which contains following information:
$D : \backslash\backslash Experiment\backslash\backslash BanglaFemeleTrainingVoice\backslash\backslash barishal\_soma$
$\backslash\backslash genoutput\backslash\backslash Recorded\_Audio\_000.txt$
$D : \backslash\backslash Experiment\backslash\backslash BanglaFemeleTrainingVoice\backslash\backslash barishal\_soma$
$\backslash\backslash genoutput\backslash\backslash Recorded\_Audio\_001.txt$
$D : \backslash\backslash Experiment\backslash\backslash BanglaFemeleTrainingVoice\backslash\backslash barishal\_soma$

$\backslash\backslash genoutput\backslash\backslash Recorded\_Audio\_002.txt$

13. Put output_generator.cpp in the folder Neural_Network. Now execute the program. This will create the following file-

a. Output.scp which contains following files
$D:\backslash\backslash Experiment\backslash\backslash BanglaFemeleTrainingVoice\backslash\backslash barishal\_soma$
$\backslash\backslash genoutput\backslash\backslash Recorded\_Audio\_000.out$
$D:\backslash\backslash Experiment\backslash\backslash BanglaFemeleTrainingVoice\backslash\backslash barishal\_soma$
$\backslash\backslash genoutput\backslash\backslash Recorded\_Audio\_001.out$
$D:\backslash\backslash Experiment\backslash\backslash BanglaFemeleTrainingVoice\backslash\backslash barishal\_soma$
$\backslash\backslash genoutput\backslash\backslash Recorded\_Audio\_002.out$

14. Put Integrate_MLN.cpp in the folder Neural_Network. Execute the program with epoch=300. Choose 1 from the menu which means training stage. It will take 7-10 days or more to execute based on configuration of the PC using.

15. Now testing stage. Choose 2 from the menu which means testing. It is actually close test.

16. Put log_converter.cpp in the folder Neural_Network. Make sure Input file = TrainOutput.scp. Execute the program which will create the following file-
TrainLogOutput.scp which contains following information-

$D:\backslash\backslash Experiment\backslash\backslash BanglaFemeleTrainingVoice\backslash\backslash barishal\_soma$
$\backslash\backslash genoutput\backslash\backslash Recorded\_Audio\_000.log$
$D:\backslash\backslash Experiment\backslash\backslash BanglaFemeleTrainingVoice\backslash\backslash barishal\_soma$
$\backslash\backslash genoutput\backslash\backslash Recorded\_Audio\_001.log$
$D:\backslash\backslash Experiment\backslash\backslash BanglaFemeleTrainingVoice\backslash\backslash barishal\_soma$
$\backslash\backslash genoutput\backslash\backslash Recorded\_Audio\_002.log$

17. Compare some .out and corresponding .log files to check if the Neural Network could recognize the data or not.

18. Make a folder Converted (any name can be used) into the folder Experiment. Put convert.c into the folder Converted. Also put TrainLogOutput.scp which was created in the last step into the folder Neural_Network.

19. Make another folder Train (you can use any name) into the folder Converted. Now open and run convert.c using visual studio software. After executing this program the following files will be generated:
a. Out_vec.scp will be generating into the folder Converted. In this file contains the following information
$Train\backslash file1.vec$
$Train\backslash file2.vec$
$Train\backslash file3.vec$
b. file1.vec, file2.vec, file3.vec files will be created into the folder Train.

20. Copy and paste Out_vec.scp into the folder converted then rename this file named Train.scp.

21. Make a folder HMMTraining into the folder Experiment. HMMTraining folder must contains the following files:
$proto\_5states\_39dim.txt(calledinitialmodelofHMM)$
$Train.scp(gettingformthelaststep.nowopenTrain.scpand$
$gotothelastlineandpressenterandfinallysaveit)$
$HTK\_39dpf.config$
$HCompV.exe$

22. Make a folder named hmm0 (make another folder model into the folder hmm0) into the folder HMMTraining.

23. Copy Train folder from the converted folder and paste it into the folder HMMTraining.

24. Run the following command

$HCompV - A - D - T1 - CHTK\_39dpf.config - STrain.scp - m - v0.01 - f0.01 -$
$Mhmm0\backslash model\backslash$
$proto\_5states\_39dim.txt$

25. After executing the above command the following files will be generated into the folder model.
a. proto_5states_39dim (it can be renamed as .cpp and see the data to understand)

b. vFloors

26. open HMMTraining -> hmm0 -> model then copy and paste proto_5states_39dim and rename it as hmmdefs.

27. Copy monophones0 and paste it into the folder model. This file contains 51 words. From this file remove sp and then save it.

28. Now open hmmdefs and copy the codes from h ”proto_5states_39dim” (4th line) to $<$ ENDHMM $>$ (last line) and paste it 50 times. So total will be 51.

29. Now remove h ”proto_5states_39dim” (51 times) and write aa, ch, (total 51 word)

30. Open HMMTraining -> hmm0 -> model then copy and paste vFloors and rename it as macros.

31. Now copy the following from proto_5state_39dim and paste it in macros (at the top).
$\sim$ o
>STREAMINFO-> 1 39
> VECSIZE> 39>NULLD><USER><DIAGC>

32. Make a folder hmm1 into the folder HMMTraining. HMMTraining must contains the following files:
a. HERest.exe
b. phones0.mlf

33. Now run following command.

$$HERest-A-D-T1-CHTK\_39dpf.config-Iphones0.mlf-t250.0150.01000.0-STrain.scp-Hhmm0\backslash model\backslash macros-Hhmm0\backslash model\backslash hmmdefs-Mhmm1hmm0\backslash model\backslash monophones0$$

34. After executing the above command the following files will be generated into the folder hmm1.
a. hmmdefs

b. macros

35. Make another folder hmm2 into the folder HMMTraining and run the following command.
$HERest - A - D - T1 - CHTK\_39dpf.config - Iphones0.mlf - t250.0150.01000.0 -$
$STrain.scp - Hhmm1\backslash$
$macros - Hhmm1\backslash hmmdefs - Mhmm2hmm0\backslash model\backslash monophones0$

36. After executing the above command the following files will be generated into the folder hmm2.
a. hmmdefs
b. macros

37. Make another folder hmm3 into the folder HMMTraining and run the following command
$HERest - A - D - T1 - CHTK\_39dpf.config - Iphones0.mlf - t250.0150.01000.0 -$
$STrain.scp - Hhmm2\backslash$
$macros - Hhmm2\backslash hmmdefs - Mhmm3hmm0\backslash model\backslash monophones0$

38. After executing the above command the following files will be generated into the folder hmm3.
a. hmmdefs
b. macros

39. Open practice -> HMMTraining -> hmm4 (create this folder) -> copy hmmdefs and macros form hmm3 and paste those into the folder hmm4

40. Now open hmmdefs and copy form h ”sil” to <ENDHMM> and paste it at the end and rename h ”sil” into h ”sp” and also need some necessary changes

41. hmm4 folder must contain the following information:
a. HHEd.exe
b. Sil.hed

42. Copy monophones0 (from hmm0 -> model) and paste it into the folder hmm4. Now rename monophones0 into monophones1 and after that open it and write sp at the end of the

file (remove blank line from the last).

43. Make a folder hmm5 into the folder HMMTraining.

44. Now run the following command
$HHEd - A - D - T1 - Hmacros - Hhmmdefs - M..\backslash hmm5sil.hedmonophones1$
$(putHHEDintofolderhmm4)$
or
$HHEd-A-D-T1-Hhmm4\backslash macros-Hhmm4\backslash hmmdefs-Mhmm5hmm4\backslash sil.hedhmm4\backslash$
$monophones1(HHEDshouldbeputintoHMMTrainingfolderinstedofhmm4)$

45. After executing this command the following files will be generated into the folder hmm5.
a. hmmdefs
b. macros

46. copy HTK_39dpf.config and Train.scp and paste it into the folder hmm5.

47. Make a folder hmm6 into the folder HMMTraining. At that time HMMTraining folder must contains the following files:
a. HERest.exe
b. monophones1
c. phones1.mlf

48. Now run the following command
$HERest - A - D - T1 - CHTK\_39dpf.config - Iphones1.mlf - t250.0150.03000.0 -$
$STrain.scp - Hhmm5\backslash$
$macros - Hhmm5\backslash hmmdefs - Mhmm6monophones1$

49. After executing this command the following files will be generated into the folder hmm6.
a. hmmdefs
b. macros

50. Make a folder hmm7 into the folder HMMTraining.

51. Now run the following command

$HERest - A - D - T1 - CHTK\_39dpf.config - Iphones1.mlf - t250.0150.03000.0 -$
$STrain.scp - Hhmm6\backslash$
$macros - Hhmm6\backslash hmmdefs - Mhmm7monophones1$

52. After executing this command the following files will be generated into the folder hmm7.
a. hmmdefs
b. macros

53. put HVite.exe, dict, words.mlf into the folder HMMTraining.

54. Now run the following command.
$HVite - A - D - T1 - l * -oSWT - bSENT - END - CHTK\_39dpf.config -$
$Hhmm7\backslash macros - Hhmm7\backslash$
$hmmdefs - ialigned.mlf - m - t250.0150.01000.0 - ylab - a - Iwords.mlf$
$- STrain.scpdictmonophones1 > HVite\_log.txt$

55. It will take a little bit time to execute this command. After executing this command the following files will be generated into the folder HMMTraining:
a. aligned.mlf
b. HVite_log.txt

56. Make a folder hmm8 into HMMTraining.

57. Run following command
$HERest - A - D - T1 - CHTK\_39dpf.config - Ialigned.mlf - t250.0150.03000.0 -$
$STrain.scp - Hhmm7\backslash$
$macros - Hhmm7\backslash hmmdefs - Mhmm8monophones1$

58. After executing the above command the following files will be generated into the folder hmm8.
a. hmmdefs
b. macros

59. Make a folder hmm9 into HMMTraining.

60. Run following command
$HERest - A - D - T1 - CHTK\_39dpf.config - Ialigned.mlf - t250.0150.03000.0 -$
$STrain.scp - Hhmm8\backslash$
$macros - Hhmm8\backslash hmmdefs - Mhmm9monophones1$

61. After executing this command the following files will be generated into the folder hmm9.
a. hmmdefs
b. macros

62. Make mktri.led into the HMMTraining. Open mktri.led and go to TC line then press enter and save it.

63. Put HLEd.exe into the folder HMMTraining.

64. Run the following command
$HLEd - A - D - T1 - ntriphones1 - l * -iwintri.mlfmktri.ledaligned.mlf$

65. After executing this command the following files will be generated into the folder HMM-Training:
triphones1
wintri.mlf

66. Install ActivePerl 5.14.2

67. Put maketrihed into the folder HMMTraining.

68. Run the following command
$perlF : \backslash experiment\backslash HMMTraining\backslash maketrihedF : \backslash experiment\backslash$
$HMMTraining\backslash monophones1F : \backslash experiment\backslash HMMTraining\backslash triphones1$

69. After executing this command mktri.hed will be generated into the drive C (in perl directory). Copy mktri.hed and paste it into HMMTraining folder.

70. Open monophones1 and go to the last line SP and then press enter and finally save it.

71. Open mktri.hed then change $CLF:\backslash experiment\backslash HMMTraining\backslash triphones1$ into the following format:
$CLF:\backslash\backslash experiment\backslash\backslash HMMTraining\backslash\backslash triphones1$
Create three folders named hmm10 hmm12
Change mktri.hed
$CLD:\backslash practice\backslash HMMTraining\backslash triphones1$
to
$CLD:\backslash\backslash practice\backslash\backslash HMMTraining\backslash\backslash triphones1$

Then execute the following command:
$HHEd-A-D-T1-Hhmm9\backslash hmmdefs-Mhmm10mktri.hedmonophones1$

72. The following files will be created after executing the above command
$hmm10\backslash hmmdefs$
$hmm10\backslash macros$

Next run HERest 2 more times:
$HERest-A-D-T1-CHTK_39dpf.config-Iwintri.mlf-t250.0150.03000.0-Strain.scp-Hhmm10\backslash$
$macros-Hhmm10\backslash hmmdefs-Mhmm11triphones1$

73. The files created by this command are:
$hmm11\backslash hmmdefs$
$hmm11\backslash macros$

74. Run the following command
$HERest-A-D-T1-CHTK\_39dpf.config-Iwintri.mlf-t250.0150.03000.0-sstats-Strain.scp-Hhmm11\backslash$
$macros-Hhmm11\backslash hmmdefs-Mhmm12triphones1$

75. The files created by this command are:
$hmm12\backslash hmmdefs$
$hmm12\backslash macros$
also create the following file: stats into the folder HMMTraining

76. need to put the following files into the folder HMMTraining:

$global.ded, HDMan.exe\&voxforge\_lexicon$

77. Run the following command
$HDMan-A-D-T1-bsp-nfulllist-gglobal.ded-1flogdicttrivoxforge\_lexicon$

78. After executing this command the following files will be generated:
fulllist, dicttri & flog

79. Next create a new fulllist1 file and copy the contents of the fulllist and triphones1 into it.

80. Copy the fixfulllist.pl script to our folder

81. Run the following command
$perlD:\backslash practice\backslash\ HMMTraining\backslash fixfulllist.plD:\backslash practice\backslash$
$HMMTraining\backslash fulllist1D:\backslash practice\backslash HMMTraining\backslash fulllist$

82. After executing this command the file will be created as like fulllist.

83. copy tree.hed (13kb original) into the folder HMMTraining.

84. copy mkclscript.prl into the folder HMMTraining.

85. Run the following command.
$perlD:\backslash practice\backslash\ HMMTraining\backslash mkclscript.prlTB350D:\backslash$
$practice\backslash HMMTraining\backslash monophones0D:\backslash practice\backslash HMMTraining\backslash tree.hed$

86. After executing this command the tree.hed (22kb) will be generated with additional information.

87. Then add the following codes into the end of the tree.hed
$TR1$
$AU"F:\backslash\backslash experiment\backslash\backslash HMMTraining\backslash\backslash fulllist"$
$CO"F:\backslash\backslash experiment\backslash\backslash HMMTraining\backslash\backslash tiedlist"$

$ST"F : \backslash\backslash experiment\backslash\backslash HMMTraining\backslash\backslash trees"$

88. Create three more folders named hmm13-hmm15 into the folder HMMTraining.

89. Run the following command.
$HHEd{-}A{-}D{-}T1{-}Hhmm12\backslash macros{-}Hhmm12\backslash hmmdefs{-}Mhmm13tree.hedtriphones1$

90. After executing this command tiedlist and trees will be created into the folder HMM-Training and also created hmmdefs and macros into the folder hmm13

91. Run the following command
$HERest - A - D - T1 - T1 - CHTK\_39dpf.config - Iwinter.mlf - sstats -$
$t250.0150.03000.0 - STrain.scp - Hhmm13\backslash$
$hmmdefs - Mhmm14tiedlist$

92. Run the following command
$HERest - A - D - T1 - T1 - CHTK\_39dpf.config - Iwinter.mlf - sstats - t250.0$
150.0 3000.0$- STrain.scp - Hhmm14\backslash$
$macros - Hhmm14\backslash hmmdefs - Mhmm15tiedlist$

CHAPTER 7
EXPERIMENTAL RESULT ANALYSIS

In order to achieve further insight into the performance of different HMM types the results were analyzed thoroughly. First, the overall insertion, deletion, and substitution errors were counted and the accuracy and correctness figures derived from these.

7.1 Performance figures

As explained earlier, there are three different types of recognition errors: substitution (S), deletion (D), and insertion errors (I). The total number of words is marked with N. The correctness (C) figure is calculated according to equation 8.1 and it describes the portion of words recognized correctly from all words.

$$\%WordCorrectRate = \frac{(N-S-D)}{N} \times 100\% \quad (8.1)$$

For Accuracy calculation it considers insertion (I). Equation for word accuracy can be written as follows:

$$\%WordCorrectRate = \frac{(N-S-D-I)}{N} \times 100\% \quad (8.2)$$

For the case of Sentence correct rate it considers correctly recognized sentences (H) within total number of sentences (N) to calculate sentence correct rate.

$$\%SentenceCorrectRate = \frac{(H)}{N} \times 100\% \quad (8.3)$$

7.2 Speech DataBase

At present, a real problem to do experiment on Bangla phoneme ASR is the lack of proper Bangla speech corpus. In fact, such a corpus is not available in any of the existing literature. Therefore, we develop a medium size Bangla speech corpus, which is described below.

Hundred sentences from the Bengali newspaper Prothom Alo are uttered by 30 male speakers and 30 female speakers of different regions of Bangladesh. These male sentences (30 100) are used for training corpus and these female sentences are user for training corpus. On the other hand, different 100 sentences from the same newspaper uttered by 10 different male speakers (total 1000 sentences) and different 100 sentences from the same newspaper uttered by 10 different female speakers (total 1000 sentences) are used as test corpus. All of the speakers are Bangladeshi nationals and native speakers of Bangla. The age of the speakers ranges from 20 to 40 years. We have chosen the speakers from a wide area of Bangladesh.

Recording was done in a quiet room located at United International University (UIU), Dhaka, Bangladesh. A desktop was used to record the voices using a head mounted close talking microphone. We record the voice in a place, where ceiling fan and air conditioner were switched on and some low level street or corridor noise could be heard.
Jet Audio 7.1.1.3101 software was used to record the voices. The speech was sampled at 16 kHz and quantized to 16 bit stereo coding without any compression and no filter is used on the recorded voice. For this study, stereo coding has been used in order to reduce the redundancy in stereo coding signals. One can achieve significant signal gains in stereo coding which can be utilized to either boost the quality of the reconstructed signal or to lower the bit rate whiling keeping the signal quality constant with respect to the original coder.

7.3 Experimental Setup

The frame length and frame rate are set to 25 ms and 10 ms (frame shift between two consecutive frames), respectively, to obtain acoustic features (MFCCs) from an input speech. MFCC comprised of 39 dimensional features. Then this 39 dimensional features are converted into 117 dimensional features to make it ready for Neural Network.
For designing an accurate continuous word recognizer, word correct rate (WCR) and word accuracy (WA) for test data set are evaluated using an HMM-based classifier. The train data set is used to design Bangla triphones HMMs with five states, three loops, and left-toright

Table 7.1: Word Recognition Performance for MFCC39+TRIPHONE-HMM using Male test Data set.

	MIX 1	MIX 2	MIX 4	MIX 8
H	3079	3087	3131	3023
D	37	40	34	63
S	174	163	125	204
I	23	21	8	7
N	3290	3290	3290	3290

models. Input features for the classifier are 39 dimensional MFCC. In the HMMs, the output probabilities are represented in the form of Gaussian mixtures, and diagonal matrices are used. The mixture components are set to one, two, four and eight.

To obtain the WCR and WA we have designed the following experiments for male , female and male+female test data sets:
(a) MFCC39+Triphone-HMM.
(b) MFCC38+Triphone-HMM.
(c) MFCC39+Neural Network+Triphone-HMM.

*** Experiments results are given in tables and charts where
H=Correctly Recognized,
D=Deletion,
S=Substitution,
I=Insertion,
N=Total,

7.4 Experimental Results and Discussion

The table 7.3 shows that in mix_1 the recognizer recognized 3111 words out of 3290 words perfectly. In mix_2 3104, mix_4 3111 and mix_8 3119 words recognized perfectly. In mix_1 it also shows that 12 words deleted, 167 words substituted and 33 words inserted. We can find the best result in mix_8, where deletions, substitutions and insertions are also lesser than other mixes.
**Similar description applies for tables 7.1,7.2,7.8,7.7and7.9

The table 7.6 shows that in mix_1 the recognizer correctly recognized 937 sentences out of

Table 7.2: Word Recognition Performance for MFCC38+TRIPHONE-HMM using Male test Data set.

	MIX 1	MIX 2	MIX 4	MIX 8
H	2906	2883	2898	2736
D	62	55	52	89
S	322	352	340	465
I	61	55	59	70
N	3290	3290	3290	3290

Table 7.3: Word Recognition Performance for MFCC39+Neural Network+TRIPHONE-HMM using Male test Data set.

	MIX 1	MIX 2	MIX 4	MIX 8
H	3111	3104	3111	3119
D	12	13	11	15
S	167	173	168	156
I	33	31	30	29
N	3290	3290	3290	3290

Table 7.4: Sentence Recognition Performance for MFCC39+TRIPHONE-HMM using Male test Data set.

	MIX 1	MIX 2	MIX 4	MIX 8
H	925	930	947	913
S	75	70	53	87
N	1000	1000	1000	1000

Table 7.5: Sentence Recognition Performance for MFCC38+TRIPHONE-HMM using Male test Data set.

	MIX 1	MIX 2	MIX 4	MIX 8
H	874	865	869	818
S	126	135	131	182
N	1000	1000	1000	1000

Table 7.6: Sentence Recognition Performance for MFCC39+Neural Network+TRIPHONE-HMM using Male test Data set.

	MIX 1	MIX 2	MIX 4	MIX 8
H	937	945	953	958
S	63	55	47	42
N	1000	1000	1000	1000

Table 7.7: Word Recognition Performance for MFCC39+TRIPHONE-HMM using FeMale test Data set.

	MIX 1	MIX 2	MIX 4	MIX 8
H	2885	3017	3080	3076
D	105	64	48	53
S	300	209	162	161
I	19	16	17	11
N	3290	3290	3290	3290

1000 test sentences which were not trained in neural network. It substituted 63 sentences. The recognizer recognized 945, 953 and 958 sentences in respectively mix_2, mix_4 and mix_8. We can find the best result in mix_8. In mix_8 maximum sentences recognized rather than other mixes.

**Similar description applies for tables 7.4,7.5,7.10,7.11 and 7.12

Table 7.8: Word Recognition Performance for MFCC38+TRIPHONE-HMM using FeMale test Data set.

	MIX 1	MIX 2	MIX 4	MIX 8
H	3001	2982	2998	2932
D	30	33	37	47
S	259	275	255	311
I	58	60	52	50
N	3290	3290	3290	3290

Table 7.9: Word Recognition Performance for MFCC39+Neural Network+TRIPHONE-HMM using FeMale test Data set.

	MIX 1	MIX 2	MIX 4	MIX 8
H	3118	3143	3167	3182
D	18	14	11	13
S	154	133	112	95
I	47	43	33	27
N	3290	3290	3290	3290

Table 7.10: Sentence Recognition Performance for MFCC39+TRIPHONE-HMM using FeMale test Data set.

	MIX 1	MIX 2	MIX 4	MIX 8
H	867	909	929	926
S	133	91	71	74
N	1000	1000	1000	1000

Table 7.11: Sentence Recognition Performance for MFCC38+TRIPHONE-HMM using FeMale test Data set.

	MIX 1	MIX 2	MIX 4	MIX 8
H	898	890	900	879
S	102	110	100	121
N	1000	1000	1000	1000

Table 7.12: Sentence Recognition Performance for MFCC39+Neural Network+TRIPHONE-HMM using Male test Data set.

	MIX 1	MIX 2	MIX 4	MIX 8
H	937	945	953	958
S	63	55	47	42
N	1000	1000	1000	1000

Table 7.13: Word Recognition Performance for MFCC39+TRIPHONE-HMM using Male+FeMale test Data set.

	MIX 1	MIX 2	MIX 4	MIX 8
H	6096	6066	6048	6020
D	120	106	110	140
S	364	408	422	420
I	28	36	36	28
N	6580	6580	6580	6580

Table 7.14: Word Recognition Performance for MFCC38+TRIPHONE-HMM using Male+FeMale test Data set.

	MIX 1	MIX 2	MIX 4	MIX 8
H	6012	5950	5966	5886
D	62	74	78	78
S	506	556	536	616
I	128	132	116	128
N	6580	6580	6580	6580

Table 7.15: Word Recognition Performance for MFCC39+Neural Network+TRIPHONE-HMM using Male test Data set.

	MIX 1	MIX 2	MIX 4	MIX 8
H	5912	5946	6030	5990
D	62	66	50	46
S	606	568	500	544
I	182	184	172	160
N	6580	6580	6580	6580

Table 7.16: Sentence Recognition Performance for MFCC39+TRIPHONE-HMM using Male test Data set.

	MIX 1	MIX 2	MIX 4	MIX 8
H	1836	1820	1820	1810
S	164	180	180	190
N	2000	2000	2000	2000

Table 7.17: Sentence Recognition Performance for MFCC38+TRIPHONE-HMM using Male test Data set.

	MIX 1	MIX 2	MIX 4	MIX 8
H	1810	1788	1794	1772
S	190	212	206	228
N	2000	2000	2000	2000

Table 7.18: Sentence Recognition Performance for MFCC39+Neural Network+TRIPHONE-HMM using Male test Data set.

	MIX 1	MIX 2	MIX 4	MIX 8
H	1756	1772	1796	1784
S	244	228	204	216
N	2000	2000	2000	2000

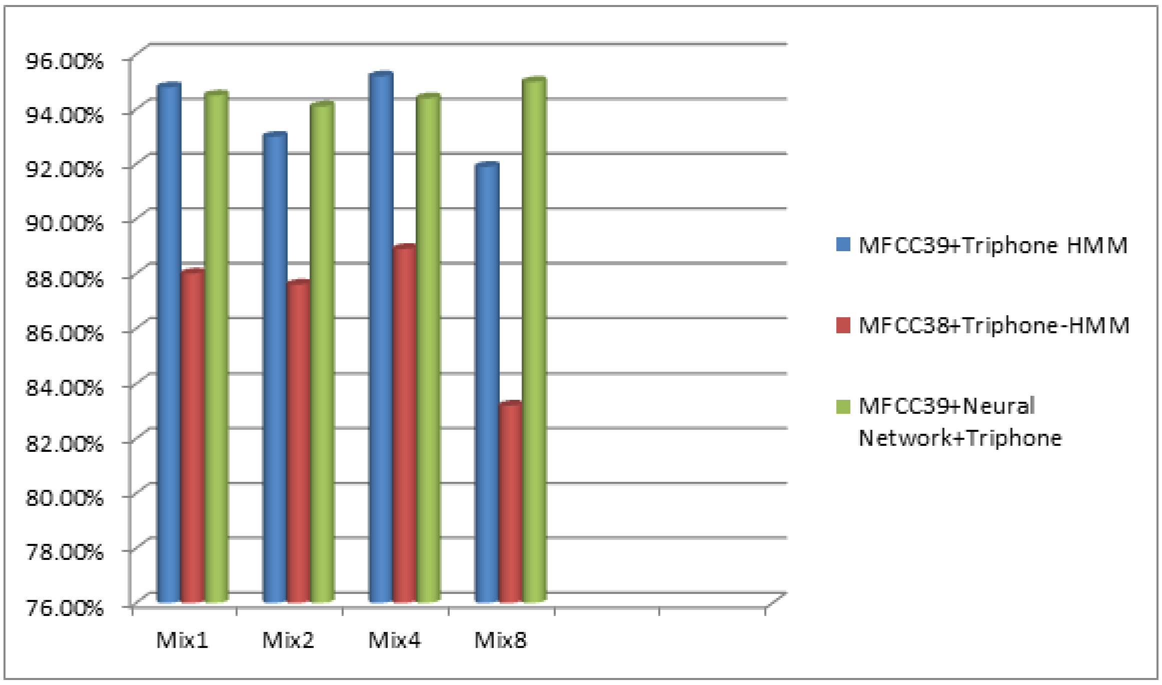


Figure 7.1: Word Correct Rate for male test data set

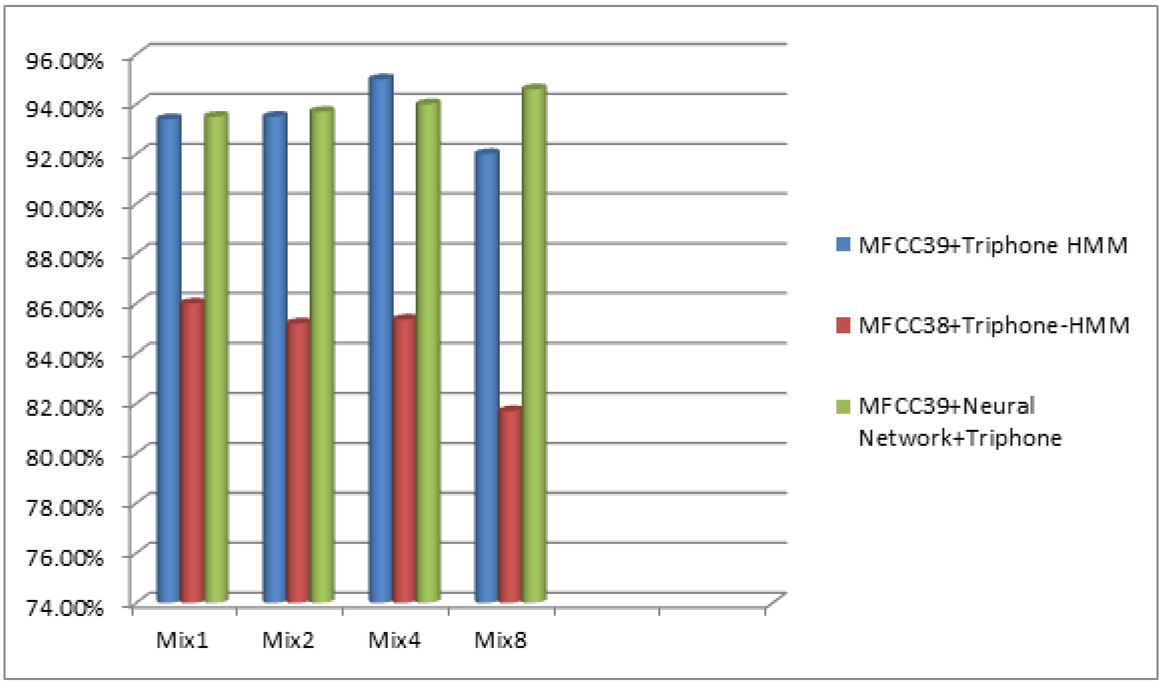


Figure 7.2: Word Accuracy for male test data set

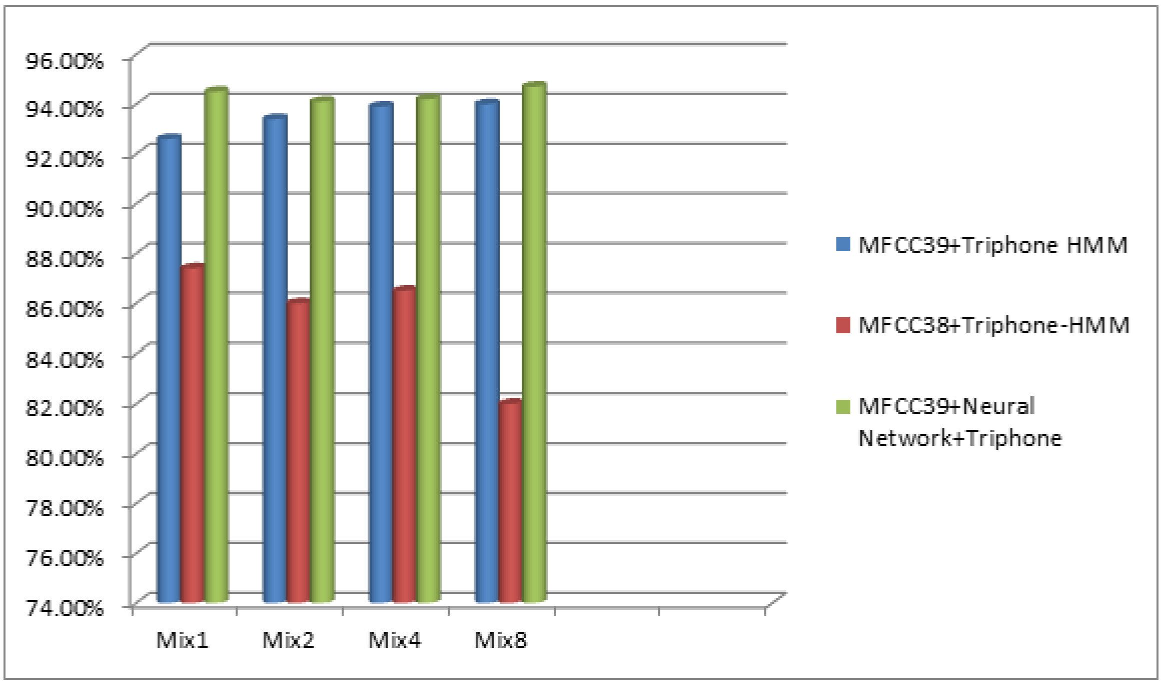


Figure 7.3: Sentence Correct rate for male test data set

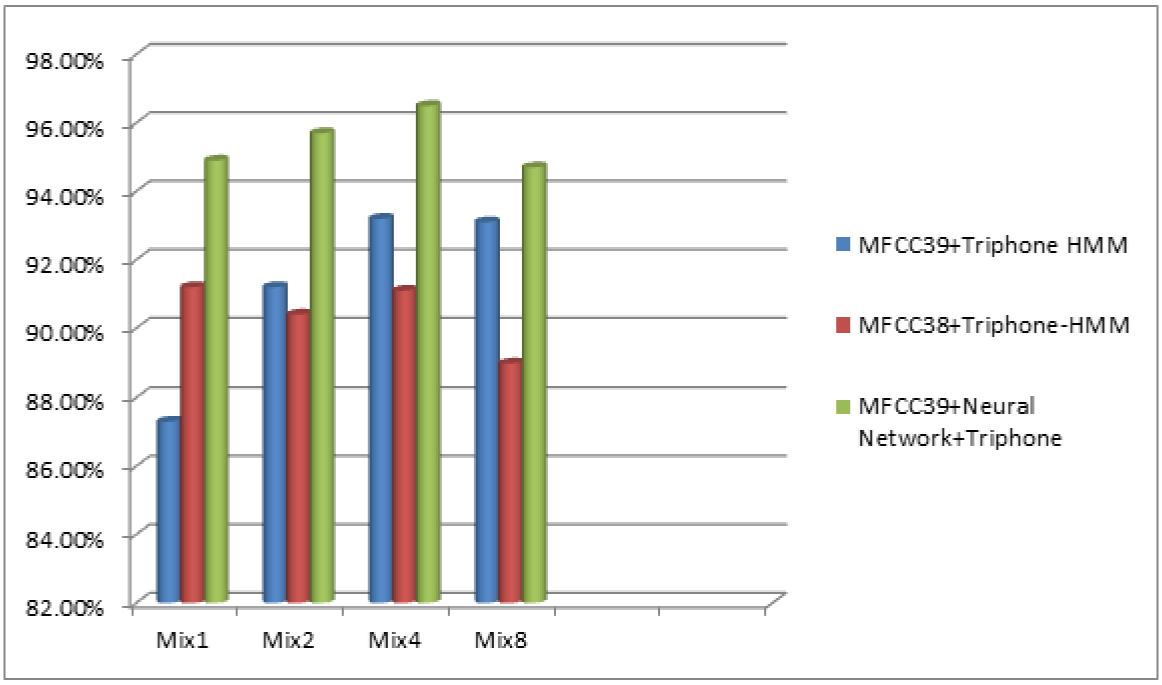


Figure 7.4: Word Correct Rate for female test data set

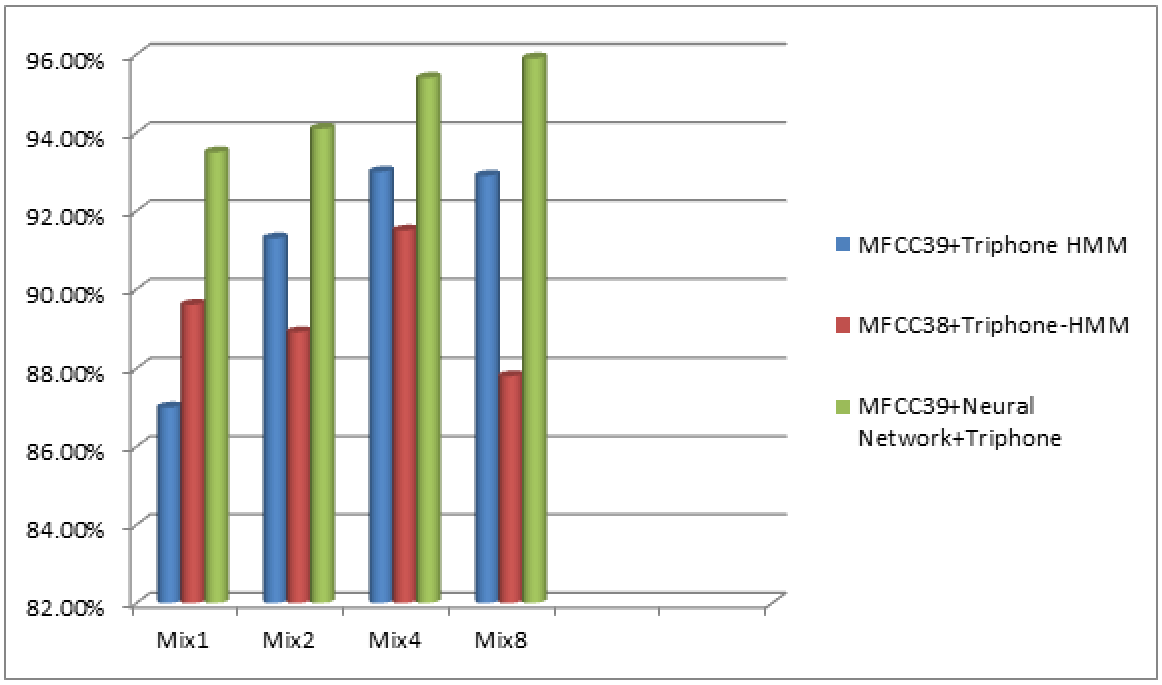


Figure 7.5: Word Accuracy for female test data set

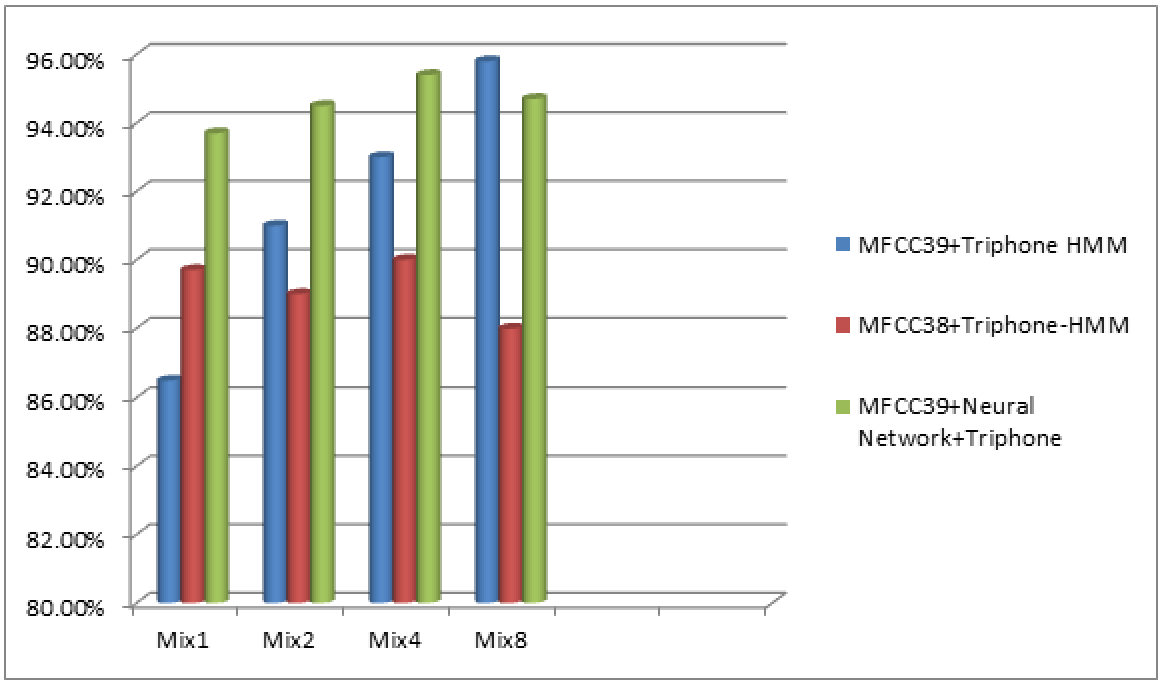


Figure 7.6: Sentence Correct Rate for female test data set

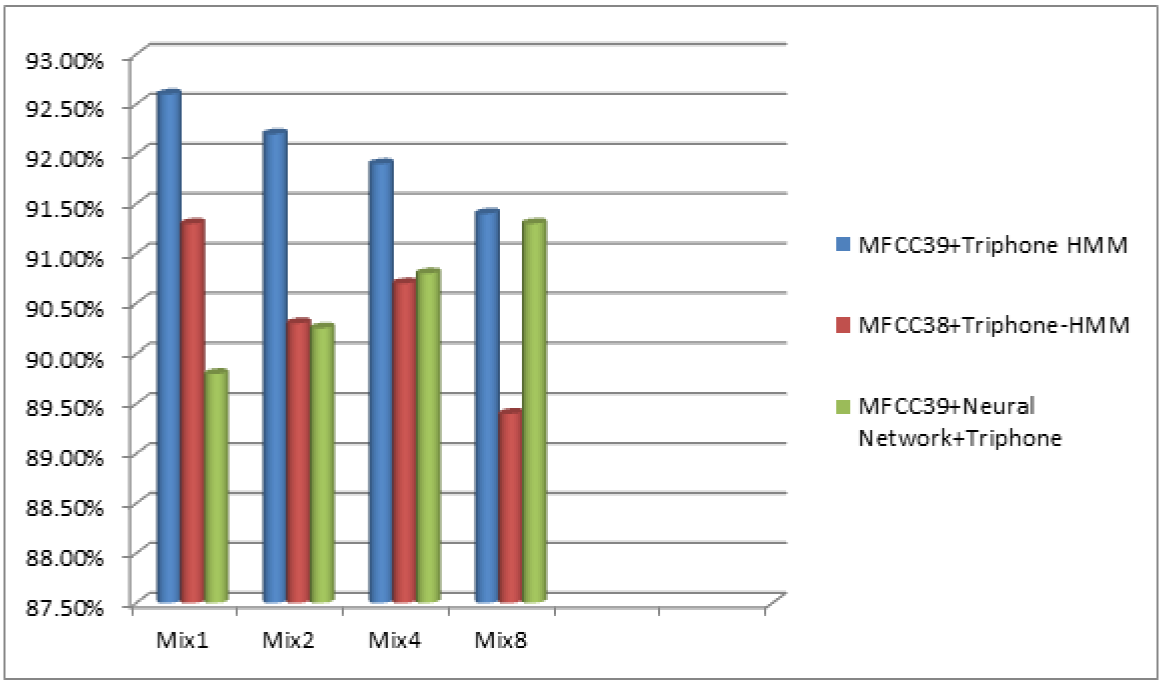


Figure 7.7: Word Correct Rate for male+female test data set

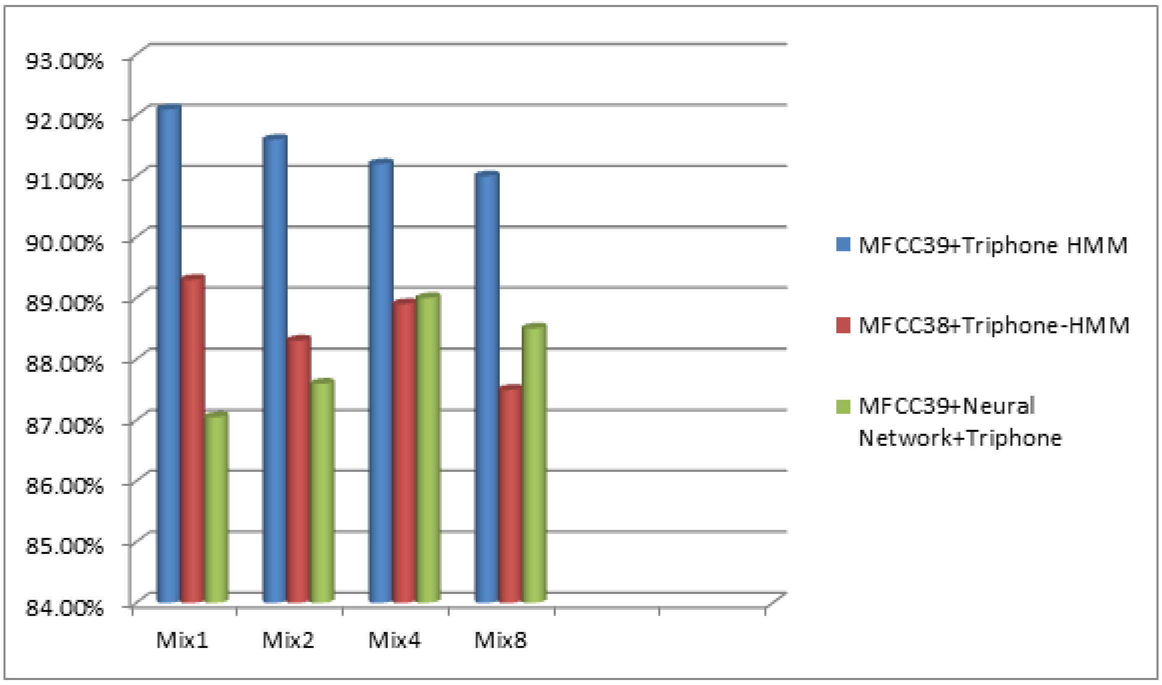


Figure 7.8: Word Accuracy for male+female test data set

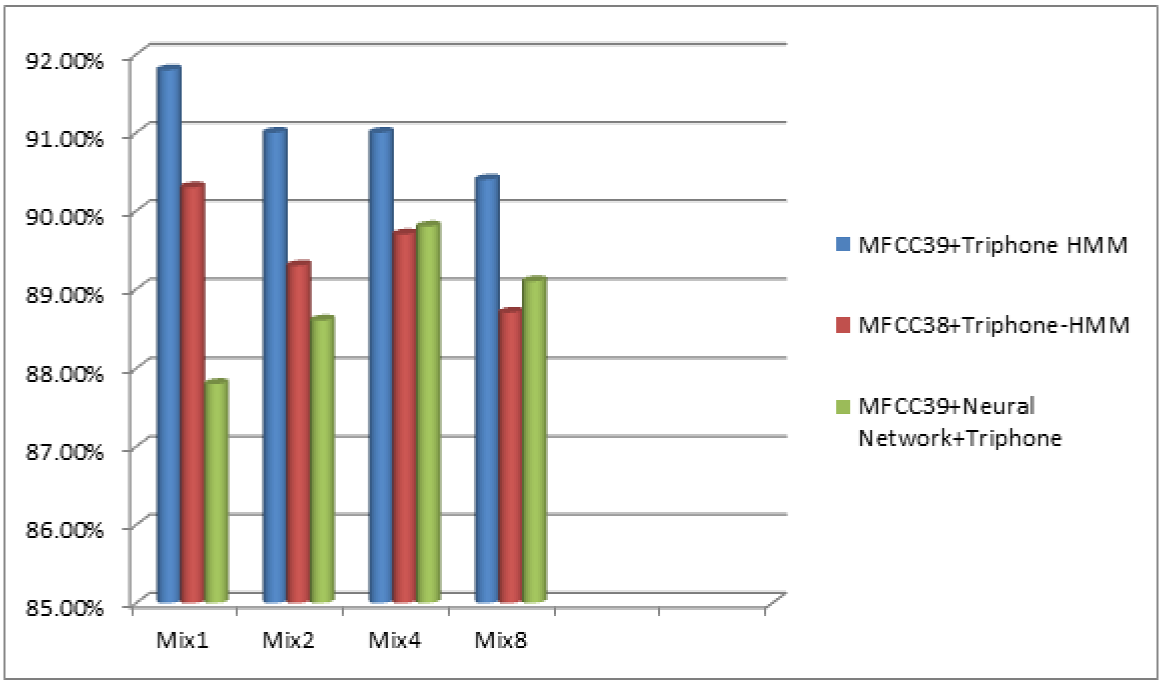


Figure 7.9: Sentence Correct Rate for male+female test data set

CHAPTER 8
CONCLUSION

The focus of this research was to show a preparation of Bangla speech corpus and to provide some experiments to obtain better word recognition performance through neural network. This covered the basics of speech recognition up to the principles of continuous speech recognition. The following conclusions are drawn from the experiments:

i) The MFCC39+Neural Network+Triphone based system provides tremendous improvement of Bangla word recognition accuracy for both training and test data.

ii) A higher Bangla word correct rate for training and test data is also obtained by the LF-based system.

iii) We have learned how the basic algorithm for isolated word recognition with HMMs works.

iv) We have learned how we can integrate additional knowledge sources into the recognition process.

v) K-mean clustering instead of model clustering. This would lead the clustering process even more towards the data-driven direction.

vi) We have learned how we can reduce the computational complexity of the search algorithm so that we are able to meet real time constraints.

Speaker independency is a major fact in many experimental applications. The methodspresented in this thesis could also be applied to such a recognizer, but speaker independency poses many new problems, as well. The need for training data increases dramatically. As there exists very few research works for Bangla ASR system, still there prevails the need for conducting lot more research works to enrich this system. So, our research based on LF-25 in the field of Bangla ASR system is a positive approach to achieve better word accuracy as well as word correct rate which will be highly beneficial to the future research analysis.

REFERENCES

[1] S. Kishore, A. W. Black, R. Kumar, and R. Sangal, “Experiments with unit selection speech databases for indian languages,” in *National seminar on Language Technology Tools, India. Hyderabad*, 2003.

[2] S. A. Hossain, M. L. Rahman, and F. Ahmed, “Bangla vowel characterization based on analysis by synthesis,” *Proc. WASET*, vol. 20, pp. 327–330, 2007.

[3] M. A. Hasnat, J. Mowla, M. Khan, *et al.*, “Isolated and continuous bangla speech recognition: implementation, performance and application perspective,” 2007.

[4] R. Karim, M. S. Rahman, and M. Z. Iqbal, “Recognition of spoken letters in bangla,” in *Proc. 5th International Conference on Computer and Information Technology (ICCIT02), Dhaka, Bangladesh*, 2002.

[5] A. Houque, “Bengali segmented speech recognition system,” *Undergraduate thesis, BRAC University, Bangladesh*, 2006.

[6] K. Rahman, M. Hossain, D. Das, T. Islam, and M. Ali, “Continuous bangla speech recognition system,” in *Proc. 6th International Conference on Computer and Information Technology (ICCIT03), Dhaka, Bangladesh*, 2003.

[7] S. Hossain, M. Rahman, F. Ahmed, and M. Dewan, “Bangla speech synthesis, analysis, and recognition: an overview,” *Proc. NCCPB*, 2004.

[8] K. Davis, R. Biddulph, and S. Balashek, “Automatic recognition of spoken digits,” *The Journal of the Acoustical Society of America*, vol. 24, no. 6, pp. 637–642, 1952.

[9] L. Bahl, P. Brown, P. De Souza, and M. Picheny, “Acoustic markov models used in the tangora speech recognition system,” in *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, pp. 497–500, IEEE, 1988.

[10] O. Fujimura, “An analysis of english syllables as cores and affixes,” *STUF-Language Typology and Universals*, vol. 32, no. 1-6, pp. 471–476, 1979.

[11] N. Dixon and H. F. Silverman, “The 1976 modular acoustic processor (map),” *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 25, no. 5, pp. 367–379, 1977.

[12] R. Schwartz, J. Klovstad, J. Makhoul, and J. Sorensen, “A preliminary design of a phonetic vocoder based on a diphone model,” in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP’80.*, vol. 5, pp. 32–35, IEEE, 1980.

[13] R. Schwartz, Y. Chow, O. Kimball, S. Roucos, M. Krasner, and J. Makhoul, "Context-dependent modeling for acoustic-phonetic recognition of continuous speech," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'85.*, vol. 10, pp. 1205–1208, IEEE, 1985.

[14] K.-F. Lee, "Context-dependent phonetic hidden markov models for speaker-independent continuous speech recognition," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 38, no. 4, pp. 599–609, 1990.

[15] S. J. Young, J. Odell, and P. C. Woodland, "Tree-based state tying for high accuracy acoustic modelling," in *Proceedings of the workshop on Human Language Technology*, pp. 307–312, Association for Computational Linguistics, 1994.

[16] A.-M. Derouault, "Context-dependent phonetic markov models for large vocabulary speech recognition," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'87.*, vol. 12, pp. 360–363, IEEE, 1987.

[17] L. Deng, M. Lennig, V. Gupta, and P. Mermelstein, "Modeling acoustic-phonetic detail in an hmm-based large vocabulary speech recognizer," in *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, pp. 509–512, IEEE, 1988.

[18] J. Ming and F. J. Smith, "Improved phone recognition using bayesian triphone models," in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, vol. 1, pp. 409–412, IEEE, 1998.

[19] R. Thangarajan, A. Natarajan, and M. Selvam, "Word and triphone based approaches in continuous speech recognition for tamil language," *WSEAS transactions on signal processing*, vol. 4, no. 3, pp. 76–86, 2008.

[20] J. Matoušek, Z. Hanzlíček, and D. Tihelka, "Hybrid syllable/triphone speech synthesis," 2005.

[21] S. Dupont, C. Ris, L. Couvreur, and J.-M. Boite, "A study of implicit and explicit modeling of coarticulation and pronunciation variation.," in *INTERSPEECH*, pp. 1353–1356, 2005.

[22] D. Jurafsky, W. Ward, Z. Banping, K. Herold, Y. Xiuyang, and Z. Sen, "What kind of pronunciation variation is hard for triphones to model?," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, vol. 1, pp. 577–580, IEEE, 2001.

APPENDIX A
CODES

A.1 Line Cutter

We use this code to remove the unnecessary heading of the MFCC data. . .

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<iostream>
using namespace std;
FILE *ifp, *ofp, *inputt, *outputt, *script;
float t[100];
int dimension=39;
double data[1000][1000];

int main()
{
    int percent=0;
    inputt=fopen("dest.mfc","r");//from where to read
    outputt=fopen("dest_out.mfc","r");//read utl where to write
    script=fopen("input.scp","w");//typically output.txt and input.scp

    int i=0;
    char inp[65];
    char outp[69];
    while(!feof(inputt))
    {
        percent++;
        fscanf(inputt, "%[^\n]%*c", &inp);
        printf("%s\tSerial:%d\n",inp,percent);
        ifp = fopen(inp, "r");
        rewind(ifp);
        fscanf(outputt, "%[^\n]%*c", &outp);
        ofp=NULL;
        if (ifp == NULL)
        {
            continue;
        }
        ofp = fopen(outp, "w");
        rewind(ofp);
        fprintf(script,"%s\n", outp);
        int counter=0;
        int last;
        float tmp;
        char line[100000];
        fscanf(ifp, "%[^\n]%*c", &line);
        fscanf(ifp, "%[^\n]%*c", &line);
        fscanf(ifp, "%[^\n]%*c", &line);
        fscanf(ifp, "%s", &line);
        fscanf(ifp, "%s", &line);

        fscanf(ifp, "%d", &last);

        fscanf(ifp, "%[^\n]%*c", &line);
        fscanf(ifp, "%[^\n]%*c", &line);
        //printf("%d\n",last);
        //while(!feof(ifp))
        int it=0;
        while(it<last)
        {
            fscanf(ifp, "%s", &line);
            for(int ii=0;ii<39;ii++)
            {
                fscanf(ifp,"%f",&tmp);
                if(ii==9 || ii==19 || ii==29 || ii==38)
                    fprintf(ofp,"%.3f\n",tmp);
                else
                    fprintf(ofp,"%.3f\t",tmp);
            }
            it++;
        }
        fclose(ifp);
        fclose(ofp);
    }
    return 0;
}
```

A.2 Matrix Format Converter

We use this code to convert the output form HList from 39 dimension to 117 dimension in order to use in our neural network…

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<iostream>
using namespace std;
FILE *ifp, *ofp, *inputt, *outputt, *script;
float t[100];
int dimension=39;
double data[1000][1000];
void out(int frame)
{
    for(int t=0;t<frame;++t)
    {
        int prev=t-3;
        int suc=t+3;
        if(prev<0)
            prev=0;
        if(suc>frame-1)
            suc=frame-1;
        for(int k=0;k<dimension;++k)
            fprintf(ofp,"%10.4lf", data[prev][k]);
        for(int k=0; k<dimension; ++k)
            fprintf(ofp,"%10.4lf", data[t][k]);
        for(int k=0; k<dimension; ++k)
            fprintf(ofp,"%10.4lf", data[suc][k]);
        fprintf(ofp,"\n");
    }
    fclose(ofp);
}
int main()
{
    int percent=0;
    inputt=fopen("input.txt","r");//from where to read
    outputt=fopen("output.txt","r");//read utl where to write
    script=fopen("input.scp","w");//typically output.txt and input.scp

    int i=0;
    char inp[65];
    char outp[69];
    while(!feof(inputt))
    {
        percent++;
        printf("%d\n",percent);
        fscanf(inputt, "%[^\n]%*c", &inp);
        ifp = fopen(inp, "r");
        rewind(ifp);
        fscanf(outputt, "%[^\n]%*c", &outp);
        ofp=NULL;
        if (ifp == NULL)
        {
            continue;
        }
        ofp = fopen(outp, "w");
        rewind(ofp);
        fprintf(script,"%s\n", outp);
        int counter=0;
        while(!feof(ifp))
        {
            int i;
            for(i=0;i<dimension;i++)
            {
                fscanf(ifp,"%lf",&data[counter][i]);
            }
            counter++;
        }
        out(counter-1);
        fclose(ifp);
    }
    return 0;
}
```

A.3 Log Creator

We use this code to convert the data from neural network for better understanding the variation…

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<iostream>
using namespace std;
FILE *ifp, *ofp, *inputt, *outputt, *script;
float t[100];
int dimension=53;
float data[1000][1000];
void out(int frame)
{
    for(int t=0;t<frame;++t)
    {
        for(int i=0;i<dimension;i++)
        {
            data[t][i] = (-10)*log10(data[t][i]);
            if(i==dimension-1)
            {
                fprintf(ofp,"%.5f\n", data[t][i]);
            }
            else
            {
                fprintf(ofp,"%.5f␣", data[t][i]);
            }
        }

    }
    fclose(ofp);
}
int main()
{
    int percent=0;
    inputt=fopen("TrainOutput.scp","r");//from where to read
    outputt=fopen("TrainOutput.output","r");//read utl where to write

    int i=0;
    char inp[65];
    char outp[69];
    while(!feof(inputt))
    {
        percent++;

        fscanf(inputt, "%[^\n]%*c", &inp);
        printf("%s\n",inp);
        ifp = fopen(inp, "r");
        rewind(ifp);
        fscanf(outputt, "%[^\n]%*c", &outp);
        ofp=NULL;
        if (ifp == NULL)
        {
            continue;
        }
        ofp = fopen(outp, "w");
        rewind(ofp);
        fprintf(script,"%s\n", outp);
        int counter=0;
        while(!feof(ifp))
        {
            int i;
            for(i=0;i<dimension;i++)
            {
                fscanf(ifp,"%f",&data[counter][i]);
            }

            counter++;
        }
        out(counter-1);
        fclose(ifp);
    }
    return 0;
}
```

A.4 HTK Environment Converter

We use this code to convert the output from the neural network to data for HTK environment…

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<string.h>

#define MAX_SAMPLE_NUM 1500000
#define MAX_FRAME_NUM 5500
#define FILTER_NUM 53
#define BIN 53
#define MAX_FNAME 512

#define IN_SCRIPT "Out_vec.test"
#define OUT_SCRIPT "Out_vec.scp"

typedef struct histogram {
        //double                **zcpa;
        //double                **d_zcpa;
        //double                **dd_zcpa;
        double          **peaks;
//      double          *power;
//      double          *d_power;
//      double          *dd_power;
        //double                **finalvector;
} HISTOGRAM;

void WriteWave(double **vec, char *fname, int fnum, int ch);

void main(void){
        int n;
        HISTOGRAM       *hist;
        char fname[MAX_FNAME]={'\0'};
        char Path[MAX_FNAME]={'\0'};
        //char LPath[MAX_FNAME]={'\0'};
        char temp[6]={'\0'};
        char tmp[10]={'\0'};
        //char phone[6]={'\0'};
        int fnum=0;
        FILE *fp,*fp1,*fp3;
        double data;
        int k,i;
        //long x,y;

        k=1;

        printf("Program is running..................\n");

    fp1=fopen(IN_SCRIPT,"r");
        if(fp1==NULL){
        printf("\nCan not open input file");
        exit(1);
        }
        rewind(fp1);

        fp3=fopen(OUT_SCRIPT,"w");
        if(fp3==NULL){
        printf("\nCan not open script file");
        exit(1);
        }
        rewind(fp3);

        if((hist=(HISTOGRAM*)malloc(sizeof(HISTOGRAM)))==NULL){
        printf("\nCan not allocate memory for histogram");
        exit(1);
    }

        memset(hist,0,sizeof(HISTOGRAM));
        if ((hist->peaks=(double **)calloc(MAX_FRAME_NUM,
                                sizeof(double *)))==NULL) {
        printf("Cant␣allocate␣memory␣for␣hist->peaks\n");
        exit(1);
        }
        for(n = 0; n < MAX_FRAME_NUM; n++){
                if ((hist->peaks[n] = (double *)calloc(BIN,
                                sizeof(double ))) == NULL){
        printf("Cant␣allocate␣memory␣for␣hist->peaks\n");
        exit(1);
                        }
        }

        while((!feof(fp1))){
                strcpy(fname,"");
                strcpy(temp,"");
                strcpy(fname,"Train\\file");
                itoa(k,temp,10);
                strcat(fname,temp);
                strcat(fname,".vec");
        putc('\n',fp3);
                fprintf(fp3,"%s",fname);
                fnum=0;

                strcpy(Path,"");
                fscanf(fp1,"%[^\n]%*c",Path);
                printf("\nInput␣File::%s",Path);

                fp=fopen(Path,"r");
                if(fp==NULL){
                                printf("\nCan␣not␣open␣input␣Data");
                                exit(1);
                }
                rewind(fp);

                n=0;
                while(!feof(fp)){
                        for(i=0;i<BIN;++i){
                          fscanf(fp,"%lf",&data);
                          hist->peaks[n][i]=data;
```

```
                    }
                    ++n;
            }
            fnum=n-1;
            WriteWave(hist->peaks,fname, fnum, BIN);

            fclose(fp);
            ++k;
    }

    fclose(fp1);
    fclose(fp3);
    return;
}

void WriteWave(double **vec, char *fname, int fnum, int ch)
{
    FILE *fp;
    int f, c, n;

    int      r_fnum;
    int      sp = 100000;
    short int sampSize;
    short int sampKind = 9;
    unsigned char *temp;
    unsigned char t1[2], t2[2], t3[2], t4[2];
    unsigned char val[15];

    unsigned char b_fnum[4], b_sp[4], b_ss[4], b_sk[4];

    //float databuf[OUTPUT_DIM];
    float databuf[FILTER_NUM];
    float **floatbuf;

    /* ?????????, allocation of temporal buffer */
    if ((floatbuf = (float **)calloc(MAX_FRAME_NUM,
                                sizeof(float *))) == NULL){
            exit(1);
    }
```

```
        for (n = 0; n < MAX_FRAME_NUM; n++){
                if((floatbuf[n]=(float*)calloc(FILTER_NUM,
                                 sizeof(float)))==NULL){
                        exit(1);
                }
        }

        /* ?????????, allocation of temporal buffer */
        if ((temp=(unsigned char*)calloc(MAX_SAMPLE_NUM*2,
                                  sizeof(unsigned char)))==NULL)
        {
                printf("Cant␣allocate␣memory␣for␣writewave\n");
                exit(1);
        }

        sampSize = ch * sizeof(float);

        /* ??????????????? */
        /* output for temporal byte swap  */
        if ((fp= fopen(fname, "wb")) == NULL){
                fprintf(stderr, "file␣open␣error\n");
                exit(1);
        }
        fwrite(&fnum, sizeof(int), 1, fp);
        fwrite(&sp, sizeof(int), 1, fp);
        fwrite(&sampSize, sizeof(short int), 1, fp);
        fwrite(&sampKind, sizeof(short int), 1, fp);
        for (f = 0; f < fnum; f++){
                for (c = 0; c < ch; c++){
                        databuf[c] = (float)vec[f][c];
                }
                fwrite(databuf, sizeof(float), ch, fp);
        }
        fclose(fp);

        /* ?????????????? */
        /* re-input for byte swap  */
        if ((fp= fopen(fname, "rb")) == NULL){
                fprintf(stderr, "file␣open␣error\n");
                exit(1);
        }
        fread(b_fnum, sizeof(unsigned char), 4, fp);
        fread(b_sp, sizeof(unsigned char), 4, fp);
        fread(b_ss, sizeof(unsigned char), 2, fp);
        fread(b_sk, sizeof(unsigned char), 2, fp);
        fread(temp, sizeof(unsigned char), MAX_SAMPLE_NUM*2, fp);

        fclose(fp);

        /* ???????????, swap of frame size */
        sprintf(t1, "%02lx", b_fnum[0]);
        sprintf(t2, "%02lx", b_fnum[1]);
        sprintf(t3, "%02lx", b_fnum[2]);
        sprintf(t4, "%02lx", b_fnum[3]);
        strcpy(val, "0x");
        strcat(val, t1);
        strcat(val, t2);
        strcat(val, t3);
        strcat(val, t4);
        sscanf(val, "%08lx", &r_fnum);

        /* ????????, swap of sampling period */
        sprintf(t1, "%02lx", b_sp[0]);
        sprintf(t2, "%02lx", b_sp[1]);
        sprintf(t3, "%02lx", b_sp[2]);
        sprintf(t4, "%02lx", b_sp[3]);
        strcpy(val, "0x");
        strcat(val, t1);
        strcat(val, t2);
        strcat(val, t3);
        strcat(val, t4);
        sscanf(val, "%08lx", &sp);

        /* ???????, swap of vector size */
        sprintf(t1, "%02lx", b_ss[0]);
        sprintf(t2, "%02lx", b_ss[1]);
        strcpy(val, "0x");
        strcat(val, t1);
```

```
	strcat(val, t2);
	sscanf(val, "%04lx", &sampSize);

	/* ?????, swap of kind of wave */
	sprintf(t1, "%02lx", b_sk[0]);
	sprintf(t2, "%02lx", b_sk[1]);
	strcpy(val, "0x");
	strcat(val, t1);
	strcat(val, t2);
	sscanf(val, "%04lx", &sampKind);


	/* ??????????, swap of feature vector */
	n = 0;
	for (f = 0; f < fnum; f++){
		for (c = 0; c < ch; c++){
		sprintf(t1, "%02lx", temp[n]);
		sprintf(t2, "%02lx", temp[n + 1]);
		sprintf(t3, "%02lx", temp[n + 2]);
		sprintf(t4, "%02lx", temp[n + 3]);
		strcpy(val, "0x");
		strcat(val, t1);
		strcat(val, t2);
		strcat(val, t3);
		strcat(val, t4);
		sscanf(val, "%08lx", &floatbuf[f][c]);
		n += 4;
		}
	}

	/* ???????????????? */
	/* output into file after byte swap */
	if ((fp= fopen(fname, "wb")) == NULL){
		fprintf(stderr, "file␣open␣error\n");
		exit(1);
	}

	fwrite(&r_fnum, sizeof(int), 1, fp);
	fwrite(&sp, sizeof(int), 1, fp);
```

```
        fwrite(&sampSize, sizeof(short int), 1, fp);
        fwrite(&sampKind, sizeof(short int), 1, fp);
        for (f = 0; f < fnum; f++){
                fwrite(floatbuf[f], sizeof(float), ch, fp);
        }

        fclose(fp);

        /* ?????????, deallocation of temporal buffer */
        for (n = 0; n < MAX_FRAME_NUM; n++){
                free(floatbuf[n]);
        }
        free(floatbuf);
        free(temp);

}
```