

Articulatory Feature Based Automatic Speech Recognition Using Neural Network

Kazi Ifrat

Id: 011132097

Kazi Israt

Id: 011132098

Imran Hossain Saimun

Id: 011132074

Fahima Akter

Id: 011132110

A thesis in the Department of Computer Science and Engineering presented
in partial fulfillment of the requirements for the Degree of
Bachelor of Science in Computer Science and Engineering



United International University

Dhaka, Bangladesh

June, 2018

Declaration

We, Kazi Ifrat, Kazi Israt, Imran Hossain Saimun, Fahima Akter, declare that this thesis titled, **“Articulatory Feature Based Automatic Speech Recognition System Using Neural Network”** and the work presented in it are our own. We confirm that:

This work was done mainly while in candidature for a BSc degree at United International University.

Where any part of this thesis has not been previously submitted for a degree or any other qualification at United International University or any other institution, this has been clearly stated.

Where we have consulted the published work of others, this is always clearly attributed.

Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely our own work.

We have acknowledged all main sources of help.

Where the thesis is based on work done by myself jointly with others, we have made clear exactly what was done by others and what we have contributed our self.

[Kazi Ifrat, 011132097, Department of CSE]

[Kazi Israt, 011132098, Department of CSE]

[Imran Hossain Saimun, 011132098, Department of CSE]

[Fahima Akter, 011132110, Department of CSE]

Certificate

I do hereby declare that the research works embodied in this thesis/project entitled **“Articulatory Feature Based Automatic Speech Recognition Using Neural Network”** is the outcome of an original work carried out by Kazi Ifrat, Kazi Israt, Imran Hossain Saimun, Fahima Akter under my supervision.

I further certify that the dissertation meets the requirements and the standard for the degree of [BSc] in Computer Science and Engineering.

Dr. Mohammad Nurul Huda
Professor and MSCSE Coordinator
United International University

Abstract

Many ASR systems based on hidden Markov model (HMM) have been developed over the last years. Most of these use mel frequency cepstral coefficient(MFCC) of the speech signal, that considers the time frequency distribution of signal energy. The main purpose of this research is to improve the performance of ASR systems by introducing articulatory information. The articulatory information describes the features of speech production rather than the features of acoustic signal. Articulatory Information is represented in term of articulatory features AF. A phoneme can easily be identified by using its unique AF set, which comprises the manner of articulation (vocalic, consonantal, continuant etc.) and place of articulation (tongue position: high, low, front, back etc.) The use of AFs in ASR had been investigated previously, and has been actively discussed in recent years. However, AF isn't widely used as features for ASR instead of MFCC because AF itself cannot provide enough performance. This thesis presents a method to extract DPFs using multilayer neural network(MLNs). Since AFs are designed after full consideration of speech production, an AF-space well represents the distances among phonemes corresponding to their articulation differences. The fact suggests that DPFs are efficient feature parameter for ASR. In the AF extractor construction, the MLN takes 39 dimensional MFCC vectors as input and outputs 22 dimensional AF vectors.

In this work a new Bangla speech corpus along with proper transcriptions has been developed; also various acoustic feature extraction methods have been investigated in order to find their effective integration into Bangla ASR system. The use of multiple acoustic features of the speech signal is considered for Bangla speech recognition. The features are usually a sequence of representative vectors that are extracted from speech signals and the classes are either words or sub word units such as phonemes.

The Bangla automatic speech recognition system, developed in this work, models the probability distribution of feature vectors using hidden Markov model (HMM) and adopts the Viterbi rule for classification. Experimental results are presented for medium database of female speech samples. The performance analysis of the individual methods are compared. It

has been found that out that our proposed methods out performs the existing standard MFCC based method.

Acknowledgement

We are grateful to Almighty Allah for his endowments for the successful completion of our thesis. Our heartiest gratitude, profound indebtedness and deep respect go to our supervisor Dr. Mohammad Nurul Huda, Professor, United International University (UIU), Dhaka, Bangladesh, for his constant supervision, affectionate guidance and great encouragement and inspiration. 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 United International University for providing their all support during the thesis work.

Table of Contents

1. Chapter 1 Introduction	1
1.1 Objective	1
1.2 Motivation for the Current Research	1
1.3 Organization of the paper.....	2
2. Chapter 2 Automatic Speech Recognition.....	3
2.1 Introduction.....	3
2.2 History.....	3
2.3 Types of Speech Recognition	4
2.4 ASR for Bangla Speech	5
2.4.1 Bangla phonemes	5
2.4.2 Monophone model	8
2.4.3 Triphone model.....	9
2.5 Acoustic Feature	10
2.5.1 Mel Frequency Cepstral Coefficient (MFCC)	10
2.6 Articulatory Feature	11
3. Chapter 3 Hidden Markov Model and the HTK	13
3.1 Choice of an Appropriate Model for Bangla ASR.....	13
3.1.1 Hidden Markov Models	13
3.1.2 Baum-Welch re-estimation	14

3.1.3 Recognition and Viterbi decoding	16
3.1.4 Model topology	17
3.2 HTK Overview	17
3.3 The Toolkit	18
3.3.1 Data preparation tools	19
3.3.2 Training tools	20
3.3.3 Recognition tools	21
3.3.4 Analysis tools	21
4. Chapter 4 Neural Network	22
4.1 History of Neural Network	22
4.2 Sigmoid Units	23
4.2.2 Example of Multilayer NN with Sigmoid Units	24
4.3 The Backpropagation Learning	25
4.3.1 Weight training Calculations:	26
5. Chapter 5 Building the recognizer	28
5.1 Data preparation	28
5.1.1 Speech Corpus Development	28
5.1.2 Recording Environment and Specification	28
5.2 Feature Extraction	28
5.2.1 MFCC Feature extraction	29

5.2.2 Articulatory Feature Extraction	33
5.3 Obtaining AF using MLN	36
5.3.1 Network Parameters	36
5.3.2 Training Network	37
5.3.3 Recognition Network	38
5.3.4 Epoch Analysis	39
5.4 Normalization.....	43
5.5 Hidden Markov Model Training	44
5.5.1 Generating phone level transcriptions for training	44
5.5.2 Generation of mono phone model.....	45
5.5.2.1 Creating prototype model	45
5.5.2.2 Creating HMM definition	46
5.5.2.3 Re-estimation	47
5.5.2.4 Fixing Silence Model	48
5.5.2.4 Re-estimation after fixing the silence model	49
5.5.3 Generation of tri phone models.....	50
5.5.3.1 MLF to LAB file conversion	50
5.5.3.2 Realigning the data.....	50
5.5.3.3 Re-estimation	51
5.5.3.4 Generate Context dependent Triphones	51

5.5.3.5 Tied states HMM	54
5.4 Testing with test data	56
6. Chapter 6 Experiments and Results	58
6.1 Performance figures	58
6.2 Experimental Setup	59
6.3 Experimental Results and Discussion.....	60
7. Chapter 7 Conclusion and Future works.....	66
7.1 Conclusion	66
7.2 Future Work	66
8. References.....	67

LIST OF TABLES

Table 2.1 Bangla Vowels and corresponding IPA	6
Table 2.2 Bangla Consonants and corresponding IPA.....	7
Table 2.3 Few Bengali Words With Their IPA.....	8
Table 2.4 Converting a mono phone declaration to a tri phone declaration	10
Table 5.1 Articulatory Feature Table	35
Table 5.2 Error rates for threshold .1	40
Table 5.3 Error rates for threshold .2	41
Table 5.4 Error rates for threshold .3	42
Table 6.1 Sentence Recognition Performance	60
Table 6.2 Word Recognition Performance.....	61
Table 6.3 Recognition Performance (%).....	62

LIST OF FIGURES

Figure 2.1 Schematic architecture for a simplified speech recognizer.	4
Figure 2.2 Segment of Monophone dictionary file	9
Figure 2.3 Block diagram of Mel Frequency Cepstral Coefficients.	11
Figure 2.4 Speech organs in human vocal tract system.	12
Figure 3.1 The Markov Generation Model	14
Figure 3.2 A left-to-right HMM with five states and three loops	17
Figure 3.3 HTK Processing Steps	19
Figure 5.1 MFCC feature extraction procedure	29
Figure 5.2 WAV file to MFC file conversion.....	30
Figure 5.3 Segment of Batch_HList.m file	31
Figure 5.4 A segment of an SHK file.....	32
Figure 5.5 Segment of a GEN File	33
Figure 5.6 Label file embedded in MLF file.....	34
Figure 5.7 AF file creation process	36
Figure 5.8 Designed Multilayer Neural Network	37
Figure 5.9 Training Multilayer Neural Network.....	38
Figure 5.10 Training Multilayer Neural Network.....	38
Figure 5.11 Bar chart of the error results for threshold .3	43

Figure 5.12 Normalizing AF values.....	44
Figure 5.13 illustrates the process	44
Figure 5.14 Generation of phone level transcription	45
Figure 5.15 HTK_39dpf.config file structure	45
Figure 5.16 Segment of updated proto_5states_39dim.....	46
Figure 5.17 Creation of hmmdef.....	47
Figure 5.18 Illustration of this step	47
Figure 5.19 Fixing the silence model.....	48
Figure 5.20 Illustration of this process.....	49
Figure 5.21 Re-estimation up to hmm7.	49
Figure 5.22 mlf to lab file generation.....	50
Figure 5.23 Triphone generation.....	52
Figure 5.24 A segment of wintri.mlf and triphones1 files	53
Figure 5.25 Tying transition matrices	54
Figure 5.26 Summary of this step	56
Figure 5.27 Summary of testing step	57
Figure 6.1 Word Correct rate comparison between MFCC-39 and AF-22	63
Figure 6.2 Word Accuracy comparison between MFCC-39 and AF-22	64
Figure 6.3 Sentence correct rate comparison between MFCC-39 and AF-22	65

Chapter 1 Introduction

Most of the largely spoken languages in the world have Automatic speech recognition (ASR) systems, but for Bangla very few research has been done. The insufficient speech corpus is one of the main difficulty while doing research in Bangla ASR. However, the existing ASR system have lower performance rate. In this research the main focus is to improve the performance rate by incorporating Articulatory feature. We'll introduce Multilayer Neural Network to obtain the Articulatory Feature.

1.1 Objective

The main objectives of this research are:

- To design a Bangla Automatic Speech Recognition System based on Articulatory feature.
- To obtain the AF feature using Multilayer Neural Network.
- To improve the performance than the existing Standard ASR system.

1.2 Motivation for the Current Research

Bangla is one of the major spoken languages, ranked seventh in the world [14].

About 220 million native and about 250 million total speakers are in Bangla, However, a very few work on Automatic Speech Recognition (ASR) for Bangla has been done, while people of other languages enjoy the blessings of ASR in their everyday life. A major difficulty in doing research in Bangla ASR is the insufficient speech corpus and also of a robust user-friendly toolkit. Therefore, ample scope is there for developing a user-friendly HMM based toolkit as well as a Bangla speech corpus of standard size.

Among the various methods Hidden Markov model (HMM) is a very popular and simple speech recognition tool. Other methods either involve lower performance or a computationally complex algorithm. Therefore applying the HMM technique in

combination with neural network to obtain articulatory feature seems to be a leap forward in Bangla speech research.

1.3 Organization of the paper

Chapter 2: A brief discussion on Automatic speech recognition

Chapter 3: Overview of Hidden Markov Model and the HTK tool.

Chapter 4: Overview of Neural network

Chapter 5: Full Procedure of building the recognizer.

Chapter 6: Illustration of the experimental results.

Chapter 7: Concludes the paper

Chapter 2 Automatic Speech Recognition

2.1 Introduction

Automatic Speech Recognition (ASR) is basically the conversion of human voice into text. It is also termed as Computer Speech Recognition. The task of speech translation is simplified if the voice of the speaker is properly recognized. The main task of most speech recognition software is to train the system to a particular speaker, which is referred to the term “Voice Recognition”. Again in an expanded sense, the procedure of enabling a computer to identify the sounds produced by human speech without being targeted at single speaker can be represented by the term Speech Recognition.

Speech recognition applications include call routing (e.g. “make a call”), voice dialing (e.g. “Call office”), data entry (e.g. entering a phone no), search (e.g. find a particular word), making documents (e.g. a city scan report), speech-to-text processing (e.g. emails) etc.

2.2 History

The first speech recognizer could understand only digits. In 1952, Bell laboratories designed the “Audrey” system that could recognize digits uttered by only one voice[25]. After 10 years, IBM introduced its “Shoebox” machine, which could recognize sixteen english words. In 1970, the U.S. Department of Defense DARPA Speech Understanding Researching(SUR) program took the innovation one step further by establishing the objective of automatic speech recognition not to be only recognition of words but to understand full speech. At 1980s, 2 different recognizer were int the market. The first one was speaker independent recognition system of words and the second one was an IBMs system, concentrating on the growth of large vocabulary speech recognition systems by Kurzweil text-to-speech program, By the last two decades, speech recognition technology has improved to be continuous speech systems with exceptionally high accuracy.

2.3 Types of Speech Recognition

Speech recognition systems can be distinguished in several different classes based on the ability of the recognizers to recognize different types of utterances. These classes are also based on the difficulties of ASR to decide when a speaker starts and end speaking.

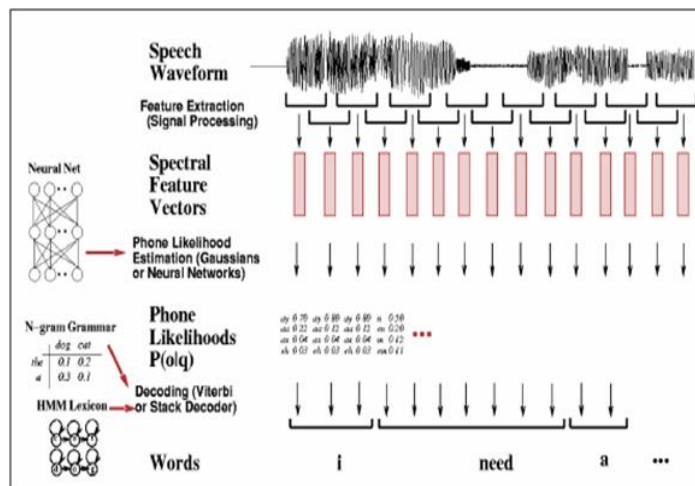


Figure 2.1 Schematic architecture for a simplified speech recognizer.

1. Isolated Words

Generally isolated word recognizers need every expression to have quite on the two sides of the window sample. That doesn't mean that it only accepts single words, but it do need a single articulation at one time. Generally, these systems be in "Listen" or "don't Listen" states, which means that they need the speaker to hold up between articulation. This class can be named as Isolated Utterance.

2. Connected Words

Connected word systems have likenesses with Isolated words, except that t different utterances have to be 'uttered together' with a short pause between them.

3. Continuous Speech

Recognizers with the capability of recognizing continuous speech are hard to create since they must use and implement special techniques to determine the limits of utterance. These type of speech recognizers gives the freedom to speak normally, while the computer decides the content. It can be named as computer dictation.

4. Spontaneous Speech

An ASR system which has the ability to recognize spontaneous speech should have the capacity to deal with various natural voice properties, for example words that run together, “hmm” and “ahs” etc.

These type of recognition can be modified by considering the impacts of the pauses while doing the recognition by:

- (1) Omit those pauses or keep those since words have to be included to the ASR system dictionary.
- (2) Increase the training data amount. This can diminish the error rate and will not increase the search complexity.

2.4 ASR for Bangla Speech

2.4.1 Bangla phonemes

There are 14 vowels in which seven are nasalized vowels, and 39 consonants. However, only 11 vowels are shown in the toddler’s book on Bangla alphabet. In Table 2.1 contains 10 vowels sounds are shown, including two more long counterpart of /ঐ/ and /ঔ/, whose IPA are /i:/ and /u:/ respectively [20]. Those 2 long vowels are articulated only from time to time differently in contrast to their short counterparts. Although Bangla has 29 consonants, there is some controversy on this number.

Letter	IPA	Symbol
অ	/ɔ/ and /o/	a
আ	/a/	aa
ই	/i/	i
ঈ	/i:/	i
উ	/u/	u
ঊ	/u:/	u
এ	/e/ and /æ/	e
ঐ	/oj/	oi
ও	/o/	o
ঔ	/ow/	ou

Table 2.1 Bangla Vowels and corresponding IPA

The consonants, which are used as a part of Bangla language, are introduced in Table 2.2.

Letter	IPA	Symbol	Letter	IPA	Symbol
ক	/k/	k	ধ	/d̪/	dh
খ	/kh/	kh	ন	/n/	n
গ	/g/	g	প	/p/	p
ঘ	/gʱ/	gh	ফ	/pʰ/	ph
ঙ	/ŋ/	ng	ব	/b/	b
চ	/tʃ/	ch	ভ	/bʱ/	bh
ছ	/tʃʰ/	chh	ম	/m/	m
জ	/dʒ/	j	য	/dʒ/	j
ঝ	/dʒʱ/	jh	র	/ɾ/	r
ট	/t/	ta	ল	/l/	l
ঠ	/tʰ/	tha	শ	/ʃ/ / /s/	s
ড	/d/	da	ষ	/ʃ/	s
ঢ	/dʱ/	dha	স	/ʃ/ / /s/	s
ণ	/n/	n	হ	/h/	h
ত	/t̪/	t	ড়	/y/	rh
থ	/t̪ʰ/	th	ঢ়	/y/	rh
দ	/d̪/	d			

Table 2.2 Bangla Consonants and corresponding IPA

Table 2.3 shows few Bengali words and their written forms with their corresponding IPA [20]. From the table, it is seen that based on the succeeding phonemes ‘ম’, ‘চ’ and ‘ব’, the same ‘আ’ (/a/) has different pronunciation. These pronunciations are sometimes short or long. For short and long ‘আ’, 2 different phonemes /ax/ and /aa/ are used respectively. Starting and

ending silence (/sil/) and short pause (/sp/) are also included in it.

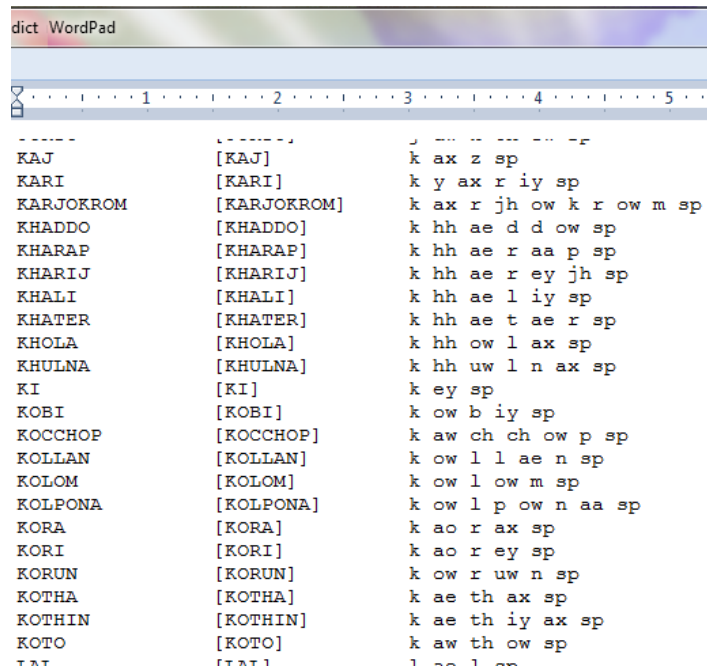
Word	Pronunciation	IPA	Symbol
আমরা	AAMRA	/a m r a/	/aa m r ax/
আগমন	AAGOMON	/a g m n/	/aa g ow m ow n/
ঢাকা	DHAAKA	/d ^h a k a/	/ dh ax k ax/

Table 2.3 Few Bengali Words With Their IPA

2.4.2 Monophone model

The articulation of a word can be represented as a sequel of symbols associated with the individual units of sound that make up a word. Those units are called 'phones' or 'phonemes'. A monophone is basically a single phoneme. Thus a monophone model for Bangla can be created from Table 2.1 and Table 2.2 along with some help from reference [22]. A silence /sil/ and short pause /sp/ monophone is added to create a silence model. An additional monophone /ax/ is added to indicate a short 'আ', where /aa/ indicates long 'আ' [20].

Considering the above mentioned facts, a dictionary file has been created manually to construct the monophone model. Figure 2.2 provides a screen shot of the dictionary file segment.



Word	Phonetic Transcription	Phoneme Sequence
KAJ	[KAJ]	k ax z sp
KARI	[KARI]	k y ax r iy sp
KARJOKROM	[KARJOKROM]	k ax r jh ow k r ow m sp
KHADD0	[KHADD0]	k hh ae d d ow sp
KHARAP	[KHARAP]	k hh ae r aa p sp
KHARIJ	[KHARIJ]	k hh ae r ey jh sp
KHALI	[KHALI]	k hh ae l iy sp
KHATER	[KHATER]	k hh ae t ae r sp
KHOLA	[KHOLA]	k hh ow l ax sp
KHULNA	[KHULNA]	k hh uw l n ax sp
KI	[KI]	k ey sp
KOBI	[KOBI]	k ow b iy sp
KOCCHOP	[KOCCHOP]	k aw ch ch ow p sp
KOLLAN	[KOLLAN]	k ow l l ae n sp
KOLOM	[KOLOM]	k ow l ow m sp
KOLPONA	[KOLPONA]	k ow l p ow n aa sp
KORA	[KORA]	k ao r ax sp
KORI	[KORI]	k ao r ey sp
KORUN	[KORUN]	k ow r uw n sp
KOTHA	[KOTHA]	k ae th ax sp
KOTHIN	[KOTHIN]	k ae th iy ax sp
KOTO	[KOTO]	k aw th ow sp

Figure 2.2 Segment of Monophone dictionary file

However, the context impact isn't considered in the phonetic plan, as it tends to give less recognition performance for an utterance which is not more regular. Therefore a context dependent phonetic scheme is needed to get better performance.

2.4.3 Triphone model

A tri phone model is derived to address the context independency of the mono phone model. A tri phone is basically a group of 3 phonemes in "L-X+R" form where the "L" phone (i.e. the left-hand phone) goes before "X" phone and the "R" phone (i.e. the right-hand phone) tails it. Table 2.4 shows the conversion of a mono phone declaration into triphone declaration for the two words "BANGLADESH" and "SHANTI"

Bangla Word	Mono phone Declaration	Tri phone Declaration
BANGLADESH	b aa ng l aa d eh sh	b+aa b-aa+ng aa-ng+l ng-l+aa l-aa+d aa-d+eh d-eh+sh eh-sh
SHANTI	sh ax n t ih	sh+ax sh-ax+n ax-n+t n-t+ih

Table 2.4 Converting a mono phone declaration to a tri phone declaration

Some bi phones at the word boundaries will be created as contexts. They will include two phones only.

2.5 Acoustic Feature

Acoustic features that are most widespread are mainly based on models of the human auditory system. Spectral amplitude compression, nonlinear (Mel) frequency scale, decreasing hearing sensitivity at lower frequencies, and large spectral integration etc are some constraining properties of human hearing and these are already integrated in the state-of-the-art acoustic features for automatic speech recognition. The Mel Frequency Cepstrum Coefficients (MFCC) feature is the most frequently used acoustic features.

2.5.1 Mel Frequency Cepstral Coefficient (MFCC)

Mel-frequency cepstral coefficients (MFCCs) are coefficients that are usually derived from a type of cepstrum representation of the sound (a nonlinear "spectrum-of-a-spectrum"). They were introduced in the 1980's by Davis and Mermelstein and have been state-of-the-art ever since. In the MFC, the frequency bands are evenly spaced on the mel scale and this estimates the human auditory

system's response more intently than the linearly spaced frequency bands that are used in the typical cepstrum. This frequency warping allows better representation of sound.

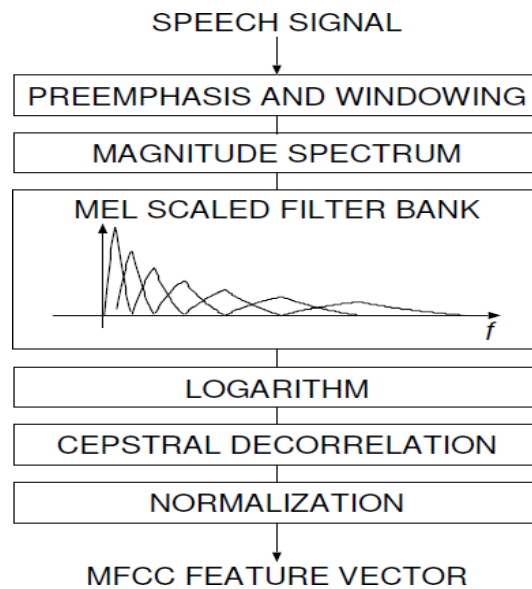


Figure 2.3 Block diagram of Mel Frequency Cepstral Coefficients.

2.6 Articulatory Feature

Articulatory feature is the property of speech sound based on its voicing or its place or manner of **articulation** in the vocal tract describing the sound.

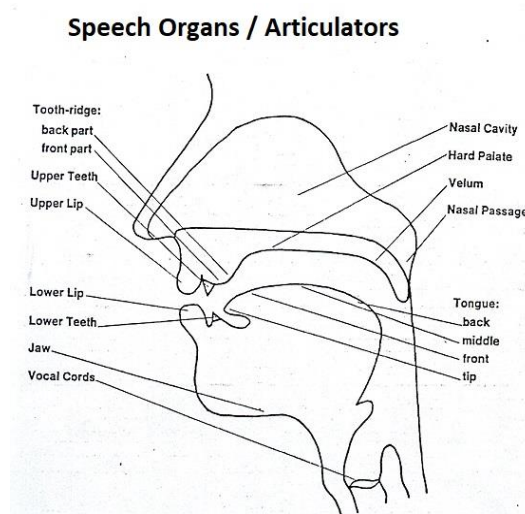


Figure 2.4 Speech organs in human vocal tract system.

The invented 22 AFs are liquid, nasal, silence, stop, fricative, short silence, bilabial, lenis, front, back, central, vowel, unvoiced, round, unround, long, short, high, low, medium, diphthong and glottal. Here, (High, Low, Medium) and (Front, Back, Central) represent tongue position in upward and downward and forward and backward directions, respectively.

Chapter 3 Hidden Markov Model and the HTK

The previous chapter introduced the Bangla speech recognition related issues and discussed the available methods for developing ASR systems. It also deliberated on the fundamentals of HMM and the various dimensions of an ASR system. This chapter gives a detailed overview of the Hidden Markov Model (HTK) and the accompanying tools used.

3.1 Choice of an Appropriate Model for Bangla ASR

In order to approximate the probability of the acoustic evidence X when a person articulates W , $P(X|W)$ acoustic models are used. To estimate such model accurately is one of the challenges in speech recognition. Because of the factors like environment, phonetic context, pronunciation, the variability of speech signal makes the approximation very complex. The very efficient acoustic modeling is done using Hidden Markov Models (HMM), which is discussed in the next section.

3.1.1 Hidden Markov Models

A hidden Markov model can be termed as a stochastic finite state automaton, that creates a sequel of observable symbols. A Markov chain is the sequence of states. The transitions between states has an associated probability named as transition probability. In order to generate an observable symbol, each state has an associated probability function. Only observation sequence is visible but the state sequence is hidden. That's the reason behind the name hidden Markov model.

In speech recognition, first the continuous speech waveform is changed into an evenly dispersed discrete parameter vector sequence. This parameter vector sequence is expected to form an correct representation of the speech wave based on the span taken by each vector. This speech waveform can be viewed as being stationary despite the fact that this is a reasonable estimation.

The task of the recognizer is to map the sequences of speech vectors and the desired underlying

sequences of symbol. This task is difficult for 2 problems. First, the mapping from symbols to speech is not balanced, as different underlying symbols can rise to similar speech sounds. Second, because of speaker variability, environment, mood etc, there are large variety in the speech waveform. Also, the limits between symbols cannot be recognized explicitly from the speech waveform. Hence, it is not possible to use the speech waveform as a sequence of concatenated static patterns.

In HMM based speech recognition, it is assumed that a Markov model generates the sequence of observed speech vectors corresponding to each word as shown in Figure 3.1. Being a finite state machine, a Markov model alters a state once in every time unit. A state (j) is added at each time (t), probability density ($b_j(o_t)$) produces a speech vector (o_t). Also, the discrete probability (a_{ij}) governs the probabilistic transition from state i to state j . This process is shown in fig. 3.1 where a model with six states generates the sequence from o_1 to o_6 .

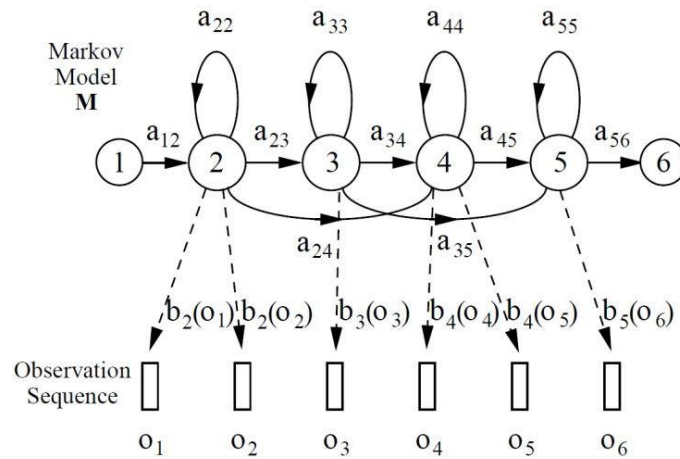


Figure 3.1 The Markov Generation Model

3.1.2 Baum-Welch re-estimation

To decide a HMM parameters, initially we make a rough guess on what the parameters would be. After the speculation is complete, more exact parameters can be built up by applying the equation Baum-Welch re-estimation.

The main issue is to approximate variances also the means of a HMM where every state of output distribution yield to be a single Gaussian component and it is

$$b_j(o_t) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_j|}} e^{-\frac{1}{2}(o_t - \mu_j)' \Sigma_j^{-1} (o_t - \mu_j)}$$

This parameter estimation would be simple, if only one state j in the HMM was present. The most extreme probability estimation of μ_j and Σ_j will be only the averages and it is

$$\hat{\mu}_j = \frac{1}{T} \sum_{t=1}^T o_t$$

$$\hat{\Sigma}_j = \frac{1}{T} \sum_{t=1}^T (o_t - \mu_j)(o_t - \mu_j)'$$

An efficient way is to at first distribute observation vectors equally among the states and then assign initial values of mean and variance for each model state. Thus using the Viterbi algorithm, we can find the maximum likelihood state sequence. Then reassign the observation vectors to model states to achieve desirable initial values. Until the estimates do not change, this process is repeated.

In this procedure, every visualization is allotted to each state in extent to the likelihood of the model being in that state when the vector was observed. Thus, if at time (t) the likelihood to be in state (j) is $L_j(t)$, then the averages shown above become the weighted averages as following

$$\hat{\mu}_j = \frac{\sum_{t=1}^T L_j(t) o_t}{\sum_{t=1}^T L_j(t)}$$

$$\hat{\Sigma}_j = \frac{\sum_{t=1}^T L_j(t) (o_t - \mu_j)(o_t - \mu_j)'}{\sum_{t=1}^T L_j(t)}$$

These are the Baum-Welch re-estimation formulas for the means and covariances of HMM

3.1.3 Recognition and Viterbi decoding

The maximum likelihood state sequence can be easily generalized to the continuous speech case if utilize base recognize. This likelihood state sequence is calculated using the same algorithm that is used in the forward probability calculation. The only difference is that a maximum operation replaces the summation. Let at time (t), $\phi_j(t)$ denote the maximum likelihood of speech vectors in state (j) for a given model (M). This partial likelihood can be computed using the recursion formula below.

$$\psi_j(t) = \max_i \{ \psi_i(t-1) + \log(a_{ij}) \} + \log(b_j(o_t))$$

If HMMs are connected together in sequence then it'll become continuous speech. The underlying symbols correspond to each HMM in the sequence. These could be either entire words for connected speech recognition or sub-words like phonemes for continuous speech recognition. The non-emitting entry and exit states are needed to join models together.

3.1.4 Model topology

Number of states and the transitions among the states are some of the issues in implementing HMMs. Again, there is no definite answer. Considering speech as a nonstationary temporal signal, a model with left-to-right topology is built to capture the temporary variability of speech. This topology has a self-transition in it and there is one transition between two adjacent states. This configures the temporal evolution of speech. A left-to-right HMM with five states and three loops is shown in Figure 3.2. This topology is the most popular HMM structure used in state-of-the-art speech recognition systems.

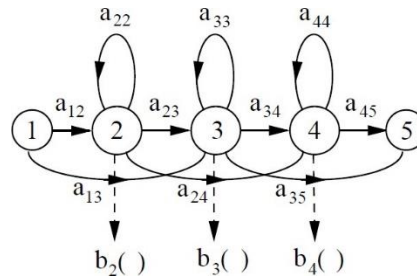


Figure 3.2 A left-to-right HMM with five states and three loops

3.2 HTK Overview

HTK is a portable and integrated suite of software toolkit to build and manipulate systems that use Hidden Markov Models. It was developed with the aim of having a general platform for research, benchmark testing and product development by the Speech Group of Cambridge University Engineering Department.

HTK is mainly designed for constructing HMM-based speech recognizers. It can be used to perform various tasks in this line like connected or isolated speech recognition using models based on a complete word or sub-word. But it is mainly appropriate for continuous speech recognition with a large vocabulary.

More than 10 library modules and 25 tools (programs) are there in the HTK. Most of the functionality of the toolkit are built into library modules. These modules make sure that the interaction between the outside world with the tools are the same and they gives the facility for creating custom tools or to integrate of recognizer functionality in an application.

HTK tools are designed to run with a traditional command-line style interface. The commands layout is the same for all the tools. Each tool has some required arguments. They also have some optional arguments that are always prefixed by a minus sign.

Most of the tools, especially editing tools, like HHed or HLEd need a script which gives direction to on what task to perform and how to edit the data given to the tool. The available scripting commands are tool specific.

3.3 The Toolkit

The tools of HTK can be understood easily if we go through the processing steps that are needed while building a continuous speech recognizer based on sub word. The processing of speech samples in HTK involves four major steps. These steps are data preparation, training, testing and analysis. These steps are shown in Figure 3.3.

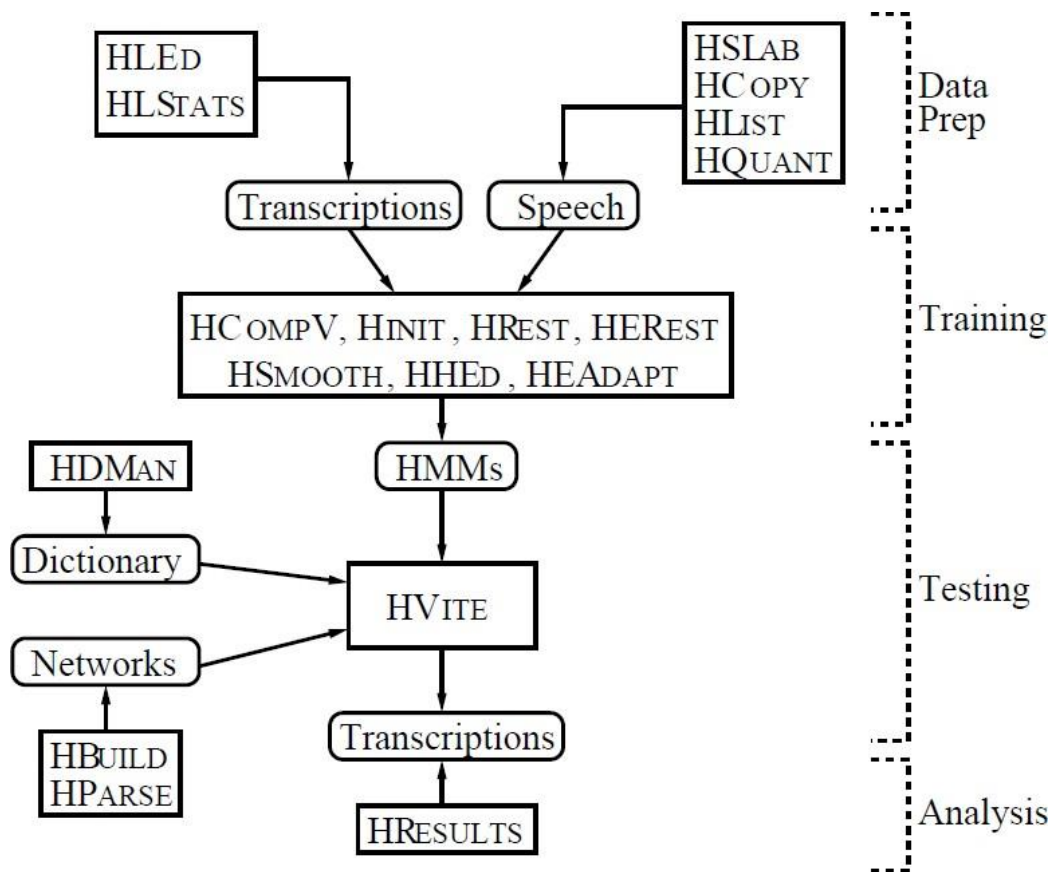


Figure 3.3 HTK Processing Steps

3.3.1 Data preparation tools

A set of speech data files and transcriptions associated with them are required to construct a set of HMMs. Before using it in training, the speech data need to be converted into an appropriate parametric form. The tool **HCOPY** is used for this. It copies multiple source files into another output file. The tool **HLIST** is used to check many speech file contents and also the results of any conversions before processing large amount of data. Transcriptions are also need to be prepared. Usually the original source label transcriptions won't have similarity with the required. Plus, HMM training may need context-dependent labels. The label editor tool **HLED** can be used to make the required

transformations in label files. This tool can also output files into a single MLF (Master Label File) which is more suitable for the further processing.

3.3.2 Training tools

The next step while constructing the recognition system is to define a prototype definition for every HMM. HTK allows any desired topology for HMMs to be built with. The training tools will compute the actual parameters later.

At first, an initial model is created. Any speech data for which the phoneme limits have been underlined, is used as bootstrap data. HInit and HRest tools give training in isolated word style by using the bootstrap data. The HMMs are created individually. The HInit tool reads the training data and marks the samples of the phoneme needed. Then by k-means procedure, it calculates an initial parameter values iteratively. The initial parameter values are further re-estimated by HRest tool. The bootstrap data is used again but this time the Baum-Welch re-estimation procedure replaces the segmental k-means procedure. A flat start can be used if no bootstrap data is available. In that case all of the phone models are initialized identically and state variances and means are same as the global speech variance and mean. HCompV tool is used for this.

After initial model's set are created, HRest tool can perform training that is embedded using the full training set. The tool executes Baum Welch re-estimation on the entire HMM phone model set concurrently. For every train articulation, the associated phone models are added together and then the algorithm forward- backward is used to accumulate the means, variances etc for every HMM.

Once all the training data is ready, the HMM parameters are re-estimated using the accumulated statistics.

HMMs should be refined incrementally while constructing a system in HTK. Thus, a set of context-independent single Gaussian phone models are build. They are then iteratively refined by including context-dependency and using multiple Gaussian distributions mixture component. The models will be cloned into context- dependent sets by the HMM definition editor tool HHed. It will apply various parameter tying and increment mixture components in specified distributions.

3.3.3 Recognition tools

HVite is the only recognition tool in HTK which utilize the token passing algorithm. To do Viterbi-based speech recognition, the tool HVite uses a set of HMMs, a network which specifies the permissible word sequences and a dictionary that defines the pronunciation of each word, as input. HVite converts the word network into a phone network and then add suitable HMM definition with each phone instance. Word networks are saved in HTK standard lattice format and can be easily created using a text-editor. Hence HTK provides two tools that can create word networks. First, HBuild creates sub-networks and uses them in high level networks. It can also generate word loops and modifies the word loop transitions to combine the bigram probabilities.

The HParse tool is introduced to convert high level grammar notation into the equivalent word network. Finally, the dictionary management tool HDMan can merge many sources and perform various transformation on each sources to construct large dictionaries.

3.3.4 Analysis tools

After the Hidden markov model based recognizer has been constructed, it is used to transcribe some pre-recorded test sentences. Then its output is compared with the correct reference transcriptions. The HResults tool is used to do the comparison. It first aligns the two transcriptions and then count insertion, deletion and substitution errors using dynamic programming.

Chapter 4 Neural Network

A neural network is special structure system build with neurons and weighted edge between those neuron which take input and produce output like pattern recognition, prediction, face detection etc.

Neural network also known as Artificial Neural Network. An Artificial Neural Network (ANN) is a data processing structure that is motivated by biological nervous system for example human's brain data processing structure. When the human's brain data processing system will implements artificially in machine or program that called artificial neural network.

4.1 History of Neural Network

The term Artificial Neural Network originated from investigation of central nervous system and their neurons, axons, dendrites and neurotransmitters which constitute the preparing components of organic neural systems examined by neuroscience. In an artificial neural system basic artificial nodes, called differently "neurons", "neurodes", "preparing components" (PEs) or "units", are associated together to build a system of nodes mimicking the natural neural systems – consequently the expression "artificial neural network"(ANN).

There is no single formal meaning of what neural system is, on the grounds that neuroscience is still brimming with questions and in light of the fact that there are numerous levels of deliberation and numerous approaches to take motivation from brain. Most would concede that it includes a system of straightforward preparing components which can show complex general nature controlled by the associations between the handling components and component parameters.

These networks are additionally like the biological neural systems as in capacities are performed on the simultaneously and in parallel by the units, as opposed to there being an unmistakable depiction of sub-tasks to which different units are appointed (see also connectionism). Right now, the term artificial neural system (ANN) has a tendency to allude generally to neural system models utilized in

insights, subjective brain research and computerized reasoning. Neural system models composed with copying of the central nervous system (CNS) as a primary concern are a subject of hypothetical neuroscience and computational neuroscience.

In present day software implementation adaptive systems is more reasonable for real-world problem solving, has very little to do with traditional artificial intelligence connectionist models. What do they have in like manner, be that as it may, is the standard of non-straight, appropriated, parallel and local preparing and adjustment.

4.2 Sigmoid Units

The unit functions take the weighted sum S as input, which indicates values coming from the connected units. The equation is

$$\sigma(S) = \frac{1}{1+e^{-S}}$$

Where $e=2.718$.

It seems like a step function while we plot the output from sigmoid units.

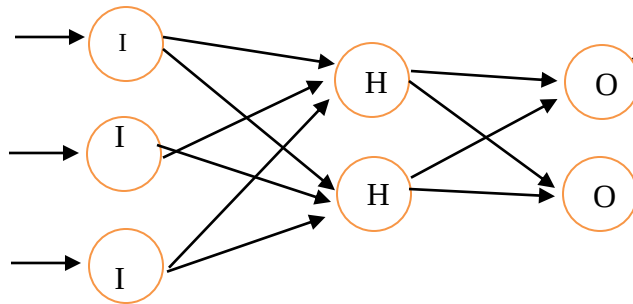
As step function is not differentiable so sigmoid function was needed. Sigmoid function is not only differentiable, but the derivative can be expressed simply in terms of the function itself:

$$\frac{d\sigma(S)}{dS} = \sigma(S) (1 - \sigma(S))$$

Note that the resultant qualities for the function extend between yet never make it to 0 and 1. This is on account of e^{-S} is never negative, and the denominator of the portion often 0 as S gets huge in the negative bearing. This inclination happens decently fast: the center ground in the vicinity of 0 and 1 is once in a while observed in view of the sharp (very close) step in the function. Because of similarity with step function, we can consider it terminating and non-terminating as in a perceptron: the output will be very near to +1 if a positive is input and the output will be very near to -1 if a negative is input. of the function itself:

4.2.2 Example of Multilayer NN with Sigmoid Units

We will now describe a neural Network which has one hidden layer, one output layer to make the back propagation learning easier. There are two sigmoid units are in the hidden layer. Initially all the units are given random weights.



Now that the perceptrons are ready, we can use the network to calculate the output from our input. We can also find out which “units” value is closer to 1 than to 0. For example, if we give the 10, 30, 20 values top to bottom into the input units. The weighted sum of H1 will be-

$$S_{H1} = (0.2 * 10) + (-0.1 * 30) + (0.4 * 20) = 2 - 3 + 8 = 7$$

Then S_{H1} is passed through the σ to give:

$$S: \sigma(S_{H1}) = 1/(1+e^{-7}) = 1/(1 + 0.000912) = 0.999 - 24 = -5$$

The weighted sum entering into H2 will be-

Then S_{H2} is passed through the σ to give:

Now it's time to calculate the weighted sum entering in the output unit O1 and that is:

the weighted sum entering in the output unit O2 will be:

$$\sigma(S_{H2}) = 1/(1+e^5) = 1/(1 + 148.4) = 0.0067$$
$$S_{O2} = (3.1 * 0.999) + (1.17 * 0.0067) = 3.1047$$

$$S_{O1} = (1.111 * 0.999) + (0.1 * 0.0067) = 1.0996$$

The output values for O1 will be

$$\sigma(S_{O1}) = 1/(1+e^{-1.0996}) = 1/(1 + 0.333) = 0.750$$

And the output values for O2 will be:

$$\sigma(S_{O2}) = 1/(1+e^{-3.1047}) = 1/(1 + 0.045) = 0.957$$

4.3 The Backpropagation Learning

The information of the network is stored in the weights. Now the main question is how to train the weights to best classify the training examples.

The backpropagation method is as below.

1. At first we design the network structure with input layer, hidden layer and output layer. In all the layer there will be perceptron with functions.
2. Then random weights usually between -0.5 to 0.5 are assigned between all the nodes.
3. Then the weights are updated using all the training examples.
4. At the end of running each epoch error is checked. In this method it is not guaranteed to find

weights that gives correct classification of training examples. So in the termination condition we have to allow some misclassification.

4.3.1 Weight training Calculations:

Since our network have more weights than perceptrons, we must first announce the system: W_{ij} to indicate the weight among units i and j . For the perceptrons, the value of Δ_{ij} will be calculated to be added to every weight after testing single example. For a particular example E And to calculate weight variations, let's start first with information on how the network would work for E . We note the target values $t_i(E)$ for O_i each output unit must generate for E . It can be noted that, because of the categorization issues $t_i(E)$ goes 0 for every units of output but for the unit associated with the correct categorization for E , $t_i(E)$ will be 1.

$$\delta H_k = h_k(E) - (1 - h_k(E)) \sum W_{ki} \delta O_i$$

Next, E propagates through the network and we can keep track of all the detected values $O_i(E)$ for the output nodes O_i . We also save all the values $h_i(E)$ for the hidden nodes. Now we

$$\delta O_k = O_k(E) - (1 - O_k(E)) (t_k(E) - O_k(E))$$

calculate the error for each output unit O_k as

$$h_k(E) - (1 - h_k(E)).$$

The error for the hidden units can be determined using the error from the output units. As we propagate this information backwards through the network this method gets its name. We calculate the error terms for each hidden unit H_k , as follows.

That means we take the error for each output unit and multiply it by the weight of the hidden unit. H_k is the output unit. Then add all these elements and multiply the sum of

After having fixed all the error values related with each unit (hidden and output), we can now shift this report into the weight changes. Δ_{ij} between units i and j . For weight W_{ij} between input unit I_i and hidden unit H_j the calculation is as follows:

$$\Delta_{ij} = \eta \delta_{H_j} X_i$$

Here, X_i is the input to the i -th input node, η is the learning rate and δ_{H_j} is the error value calculated for hidden node H_j .

For weights w_{ij} between hidden unit H_i and output unit O_j we add on:

$$\Delta_{ij} = \eta \delta_{O_j} h_i(E)$$

Here, $h_i(E)$ is the output from hidden node H_i when example E is propagated through the network, δ_{O_j} is the error value we calculated for the output node O_j .

The learning rate is used so that the weights do not change fast and thus the previous examples training is not lost.

Chapter 5 Building the recognizer

5.1 Data preparation

5.1.1 Speech Corpus Development

We build a Bangla speech corpus in medium size, which is described below.

We've selected 100 sentences from the Bangla newspaper "Prothom Alo". These sentences are spoken by 30 female speakers from different region of Bangladesh,. These sentences (total 3000 sentences) will be the Training corpus D1.

On the other hand, different 100 sentences spoken by 10 different female speakers (total 1000 sentences) from the same newspaper will be used as test corpus D2.

The speakers are Bangladeshi and they speak native Bangla. The speakers ages from 22 to 30 years. We have selected the speakers from different regions of Bangladesh like Dhaka (Central region), Cumilla - Noakhali (East region), Khulna (South-West region), Mymensing – Sylhet (North-East region), Rajshahi (West region). Though all of them speak standard Bangla, they have their regional accent.

5.1.2 Recording Environment and Specification

The speech voices recorded in a silent room of United International University (UIU), Dhaka, Bangladesh. To record the voices, a desktop was used with a head mounted close- talking microphone. Ceiling fans and ACs were remained switched on while recoding the voices. Also some corridor and street noise could be heard.

We recorded the voices using the Jet Audio software. The speech signals were quantized to 16 bit stereo coding and sampled at 16 kHz. The recorded voices aren't filtered. To avoid the redundancy in stereo coding signals we've used stereo coding.

5.2 Feature Extraction

5.2.1 MFCC Feature extraction

The first task is to compute the Mel cepstral coefficient files from the speech wav files while building the recognition system. This process involves many different stages.

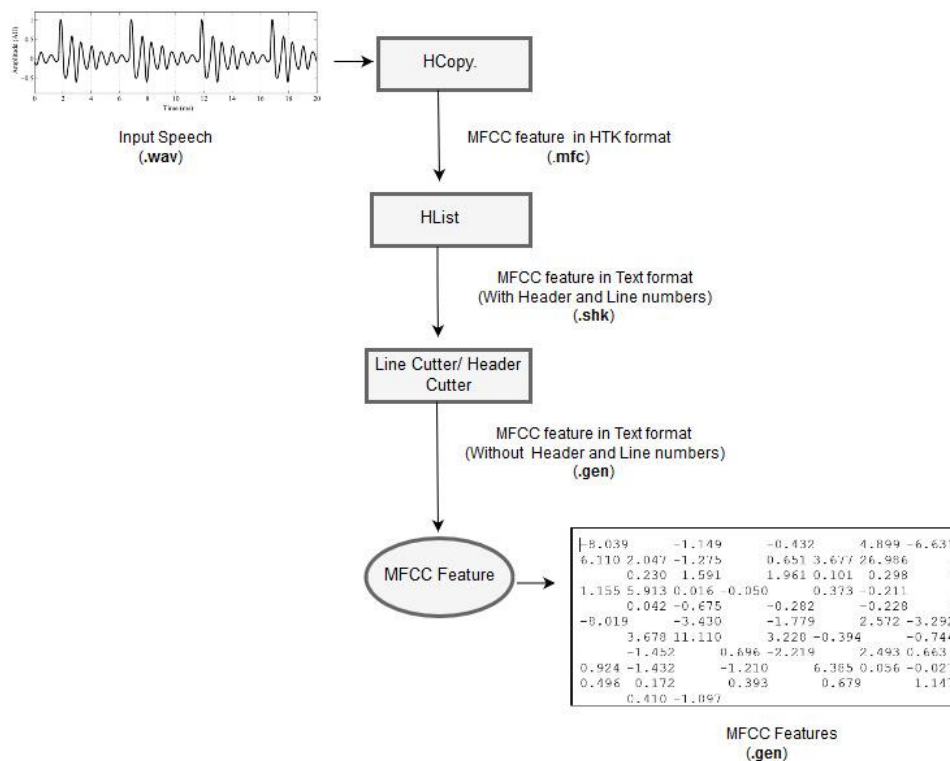


Figure 5.1 MFCC feature extraction procedure

Step 1 : WAV to MFC file conversion

WAV file is converted to MFC file using script **HCOPY**. While preparing the data the last step is to convert the speech waves into feature vectors. The tool **HCOPY** automatically converts its input to Mel frequency cepstral coefficient vectors. (**config.txt**) a configuration file is used to specify all the conversion parameters.

Step 1 is illustrated in Figure 5.2.

Command executed:

HCOPY -T 1 -C config.txt -S Train.scp

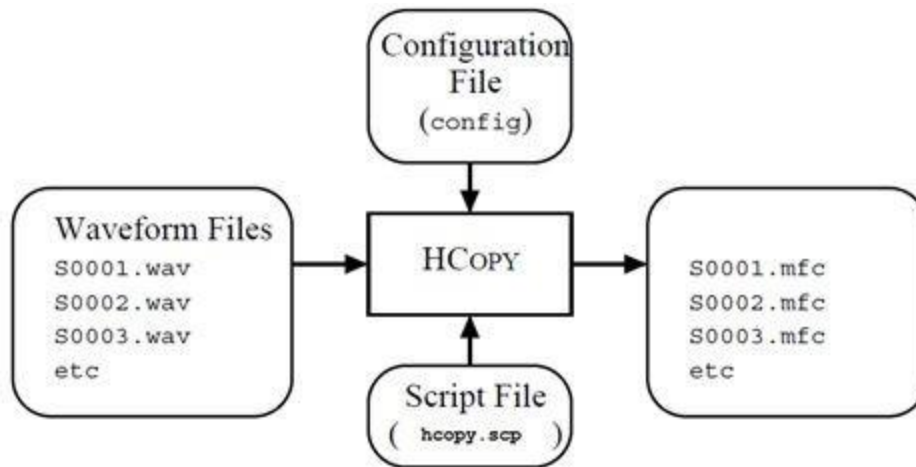


Figure 5.2 WAV file to MFC file conversion

Step 2: MFC to SHK file conversion

This step converts MFC to SHK file. HList.exe is used for that. **Batch_HList.m** is a matlab file. Figure 6.3 shows the file.

```

1 %%Input Script File
2 fin = fopen('E:\\practice\\source.mfc');
3 %%Output Script File
4 fout= fopen('E:\\practice\\dest.mfc','w');
5 %%HTK Command
6 htk='E:\\practice\\HList -h';
7 %%HList Command Generated File Extension
8 extn='.shk';
9
10 tline = fgetl(fin);
11
12 while ischar(tline)
13     disp(tline)
14     toutline=tline;
15
16     [pathstr,name,ext,versn] = fileparts(tline); % Get File Path, Name, Extension and Version
17     disp(pathstr);
18     genpath=sprintf('%s',pathstr);
19     pathstr=strcat(pathstr,'\\output'); % Insert Another Level of Folder Named output
20     genpath=strcat(genpath,'\\genoutput');
21     disp(pathstr);
22     %disp(genpath);
23     a=exist(pathstr,'dir'); %Check If The Directory Exists
24     disp(a);
25
26     if(a~=7)
27         status = mkdir(pathstr) %If Not Exist Create Directory
28         disp(status);
29     end
30

```

Figure 5.3 Segment of Batch_HList.m file

After executing Batch_HList.m, the SHK file will be generated and a segment of SHK file is shown in Figure 6.4

```

----- Source: F:\test2\mfcc\l
001.mfcc -----
  Sample Bytes:  78      Sample Kind
  Num Comps:    39      Sample Peri
  Num Samples:  198     File Format
-----

0:      -6.474   0.664  -1.451  -1.94
-1.347  -1.697  -1.284
      2.660   4.095  58.794  -0.41
-1.411   0.248   0.020
      -0.569   0.220   0.440  -0.19
0.046   0.090   0.140
      0.126   0.037  -0.130   0.06

```

Figure 5.4 A segment of an SHK file

Step 3: SHK to GEN file conversion

This step removes the header portion of SHK and save it to GEN file. A program LineCutter is used to execute the process. Figure 6.5 shows a segment of a GEN file

-8.039	-1.149	-0.432	4.899	-6.631	
6.110	2.047	-1.275	0.651	3.677	26.986
	0.230	-1.591	1.961	0.101	-0.298
1.155	5.913	0.016	-0.050	0.373	-0.211
	0.042	-0.675	-0.282	-0.228	
-8.019	-3.430	-1.779	2.572	-3.292	
	3.678	11.110	3.228	-0.394	-0.744
	-1.452	0.696	-2.219	2.493	0.663
0.924	-1.432	-1.210	6.385	0.056	-0.027
0.496	-0.172	-0.393	-0.679	-1.147	
	0.410	-1.097			

Figure 5.5 Segment of a GEN File

Step 4: GEN file to VEC file conversion

This step converts GEN to VEC file using **convert.exe**.

MFC contains header, which is not directly readable by HTK, A simple work around is to convert to human readable format, then remove header, and then convert it to HTK. format, which is .wav > .mfc > .shk > .gen > .vec. VEC files are essentially the same as MFC, but with the header portion removed.

5.2.2 Articulatory Feature Extraction

The first step of articulatory feature extraction process is to create Label files for corresponding training files.

Step 1: Labeling files

Many operations in HTK that involves speech data files requires speech that is divided into segments and a label for each segment. The labels for the speech file holds a transcription. Separate label file stores each transcription. When a large number of files is processed, label file access can be easier if

we use Master Label Files. These MLFs can be thought of as index that holds location of the main label files. The contents of a label file could be a word or phoneme. Figure 8 illustrates Label file for utterance /AALO/

```
#!MLF!#  
"*/a.lab"  
000000 300000 silB  
310000 3300000 aa  
3400000 3500000 l  
3600000 3800000 ow  
3900000 4000000 sil
```

Figure 5.6 Label file embedded in MLF file

Step 2: Articulatory Feature Table

From the Label file we get the corresponding phonemes for all the utterance. Now its time to map the phonemes with the Articulatory feature.

Following figure shows the AF table which maps each of 53 Bangla phonemes with 22 AFs.

	Silence	Short Silence	Nasal	Bilabial	Fricative	Stop	Round	Glotal
SilB	.9	.1	.1	.1	.1	.	.	.1
SilB	.9	.1	.1	.1	.1	.	.	.1
SilB	.9	.1	.1	.1	.1	.	.	.1
aa	.1	.9	.1	.9	.9	.	.	.1
aa	.1	.9	.1	.9	.9	.	.	.1
aa	.1	.9	.1	.9	.9	.	.	.1
L	.1	.1	.1	.9	.1	.	.	.1
L	.1	.1	.1	.9	.1	.	.	.1
ow	.1	.1	.1	.1	.9	.	.	.9
ow	.1	.1	.1	.1	.9	.	.	.9
ow	.1	.1	.1	.1	.9	.	.	.9
Sil	.1	.9	.1	.9	.9	.	.	.1

Figure 5.7 AF file creation process

5.3 Obtaining AF using MLN

5.3.1 Network Parameters

- ✓ Input Layer - 39 neurons
- ✓ Hidden Layers – 3
 - 1st Hidden Layer = 200 neurons
 - 2nd Hidden Layer= 100 neurons
 - 3rd Hidden Layer = 50 neurons
- ✓ Output Layer - 22 neurons
- ✓ Learning rate - .018
- ✓ Activation Function – Sigmoid
- ✓ Weight updating method – Back propagation
- ✓ Total epoch – 500

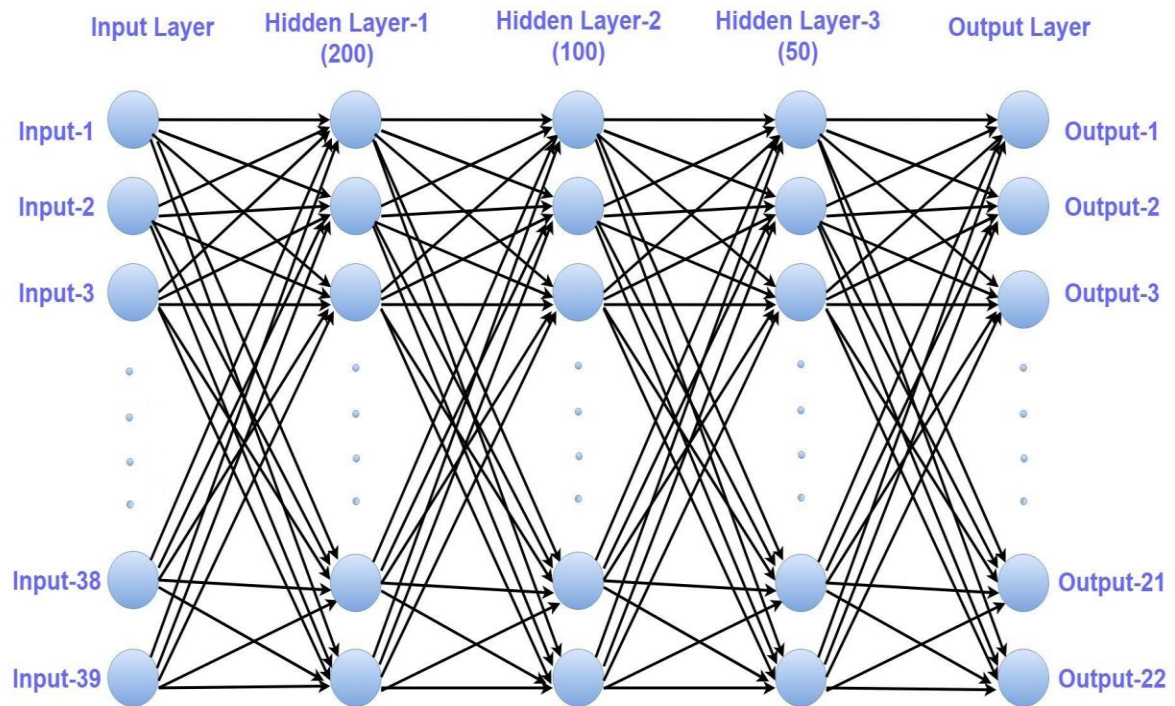


Figure 5.8 Designed Multilayer Neural Network

5.3.2 Training Network

39 dimensional MFCC features are entered into the MLN and 22 dimensional AF values are assigned as the target output. Then the network is trained for 500 epoch and the synaptic weights are updated from the initial random weights.

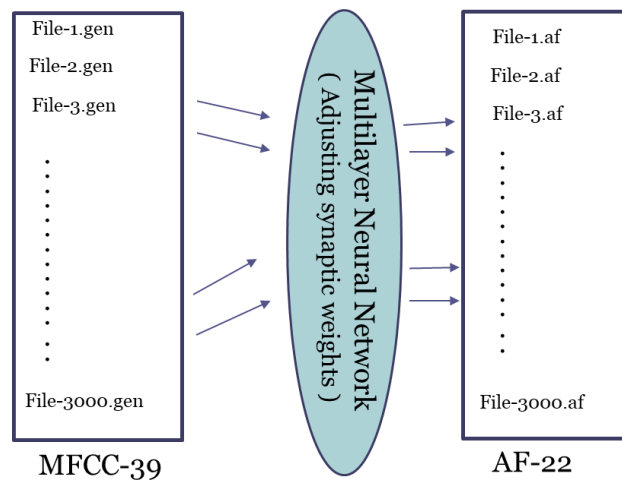


Figure 5.9 Training Multilayer Neural Network

5.3.3 Recognition Network

The trained MLN now generates 22 dimensional AF values for given 39 dimension MFCC feature. It generates 3000 Af file for each epoch.

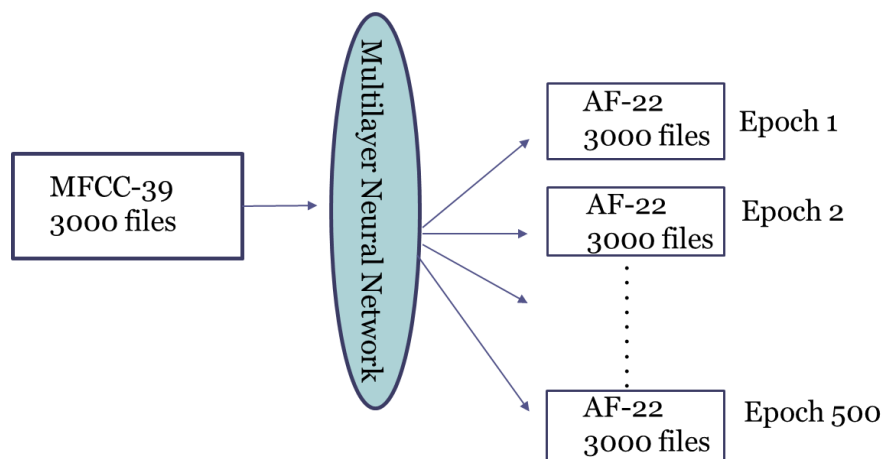


Figure 5.10 Training Multilayer Neural Network.

5.3.4 Epoch Analysis

From the 500 epoch we need to decide which one is better. We will check which epoch gives less error. For evaluating AF error rate we have considered 0.1 , 0.2 and 0.3 as threshold difference between the original AF value and MLN generated AF value to obtain better segmentation.

Error Rate:

$$\% \text{ Frame Error Rate} = (\mathbf{EF \div TF}) \times \mathbf{100\%}$$

$$\% \text{ Total AF Error Rate} = (\mathbf{EAF \div TAF}) \times \mathbf{100\%}$$

Here,

TF = Total no of Frames (926340)

EF = No of Errors in Frames

TAF = Total no of AFs (20379480)

EAF = No of Errors in AFs

Epoch	Frame Error	Frame Error Rate	AF Error	AF Error Rate	Epoch	Frame Error	Frame Error Rate	AF Error	AF Error Rate
10	676156	72.992	5468874	26.835	260	594817	64.212	4236782	20.789
20	603574	65.157	4759393	23.354	270	585852	63.244	4235174	20.782
30	601845	64.97	4658628	22.859	280	594361	64.162	4235584	20.784
40	592199	63.929	4490798	22.036	290	592121	63.92	4242024	20.815
50	607497	65.58	4679026	22.959	300	578797	62.482	4185049	20.536
60	595238	64.257	4458078	21.875	310	579064	62.511	4179873	20.51
70	594623	64.191	4453478	21.853	320	577873	62.382	4150020	20.364
80	592063	63.914	4456629	21.868	330	578592	62.46	4207866	20.648
90	587453	63.417	4406391	21.622	340	575482	62.124	4113855	20.186
100	600228	64.796	4405994	21.62	350	581487	62.773	4217778	20.696
110	595208	64.254	4394558	21.564	360	595346	64.269	4160324	20.414
120	587684	63.442	4367002	21.428	370	578281	62.426	4123632	20.234
130	587135	63.382	4279050	20.997	380	577840	62.379	4137424	20.302
140	586836	63.35	4263378	20.92	390	601656	64.95	4197925	20.599
150	581637	62.789	4330926	21.251	400	593912	64.114	4175640	20.489
160	589279	63.614	4308459	21.141	410	577420	62.333	4164790	20.436
170	613211	66.197	4297027	21.085	420	573351	61.894	4091117	20.075
180	583406	62.98	4265139	20.929	430	583095	62.946	4161433	20.42
190	591298	63.832	4230992	20.761	440	579065	62.511	4091895	20.079
200	580907	62.71	4221306	20.714	450	571843	61.731	4090302	20.071
210	576757	62.262	4225933	20.736	460	580733	62.691	4189242	20.556
220	608783	65.719	4258705	20.897	470	572153	61.765	4105161	20.144
230	574968	62.069	4175692	20.49	480	567019	61.211	4035867	19.804
240	573430	61.903	4169393	20.459	490	580717	62.689	4121812	20.225
250	584876	63.138	4170469	20.464	500	574520	62.02	4108701	20.161

Table 5.2 Error rates for threshold .1

Epoch	Frame Error	Frame Error Rate	AF Error	AF Error Rate	Epoch	Frame Error	Frame Error Rate	AF Error	AF Error Rate
10	604326	65.238	3286921	16.129	260	540401	58.337	2773474	13.609
20	573695	61.931	3213122	15.766	270	535964	57.858	2774805	13.616
30	563718	60.854	3124937	15.334	280	537338	58.007	2767544	13.58
40	546718	59.019	3002960	14.735	290	536022	57.864	2772885	13.606
50	564518	60.941	3091426	15.169	300	530476	57.266	2753139	13.509
60	549266	59.294	2963376	14.541	310	527608	56.956	2740001	13.445
70	549688	59.34	2936956	14.411	320	528783	57.083	2721425	13.354
80	548731	59.236	2940585	14.429	330	529117	57.119	2739059	13.44
90	543928	58.718	2888584	14.174	340	521728	56.321	2726081	13.377
100	549879	59.36	2923943	14.347	350	531557	57.382	2774856	13.616
110	543071	58.625	2875067	14.108	360	533996	57.646	2710672	13.301
120	541832	58.492	2892372	14.193	370	528696	57.074	2709194	13.294
130	537473	58.021	2866824	14.067	380	526465	56.833	2719891	13.346
140	540520	58.35	2801886	13.749	390	537341	58.007	2719049	13.342
150	539858	58.279	2865585	14.061	400	532308	57.464	2700747	13.252
160	540276	58.324	2846833	13.969	410	527552	56.95	2720650	13.35
170	549772	59.349	2838262	13.927	420	523543	56.517	2676296	13.132
180	535819	57.843	2814455	13.81	430	525667	56.747	2714204	13.318
190	534237	57.672	2760249	13.544	440	526298	56.815	2677228	13.137
200	532077	57.439	2774798	13.616	450	522632	56.419	2680904	13.155
210	533000	57.538	2782427	13.653	460	528939	57.1	2722742	13.36
220	545201	58.855	2796163	13.72	470	520627	56.203	2684538	13.173
230	529003	57.107	2756208	13.524	480	517012	55.812	2646364	12.985
240	529576	57.169	2750282	13.495	490	525792	56.76	2688197	13.191
250	529733	57.186	2725969	13.376	500	520577	56.197	2682099	13.161

Table 5.3 Error rates for threshold .2

Epoch	Frame Error	Frame Error Rate	AF Error	AF Error Rate	Epoch	Frame Error	Frame Error Rate	AF Error	AF Error Rate
10	552371	59.629	2376432	11.661	260	477694	51.568	1968106	9.657
20	526139	56.798	2276216	11.169	270	477317	51.527	1964086	9.638
30	517200	55.833	2203301	10.811	280	476347	51.422	1948752	9.562
40	498615	53.826	2144782	10.524	290	474921	51.269	1961490	9.625
50	516664	55.775	2192976	10.761	300	470901	50.835	1949727	9.567
60	497191	53.673	2100917	10.309	310	468113	50.534	1927605	9.459
70	490401	52.94	2063210	10.124	320	467136	50.428	1920680	9.425
80	493028	53.223	2073906	10.176	330	469173	50.648	1923926	9.441
90	487805	52.659	2042231	10.021	340	462602	49.939	1927548	9.458
100	494067	53.335	2075592	10.185	350	472899	51.05	1955944	9.598
110	486702	52.54	2026109	9.942	360	469546	50.688	1911979	9.382
120	486406	52.508	2035899	9.99	370	468321	50.556	1908691	9.366
130	484129	52.263	2039355	10.007	380	467375	50.454	1912381	9.384
140	482758	52.115	1993742	9.783	390	471940	50.947	1911642	9.38
150	479315	51.743	2008101	9.854	400	469187	50.65	1908789	9.366
160	479397	51.752	2005653	9.842	410	467438	50.461	1917118	9.407
170	486824	52.553	2010496	9.865	420	463184	50.002	1889848	9.273
180	478897	51.698	1991523	9.772	430	465192	50.218	1909580	9.37
190	472620	51.02	1954752	9.592	440	465419	50.243	1892704	9.287
200	473242	51.087	1965313	9.644	450	462994	49.981	1888445	9.266
210	478311	51.634	1989059	9.76	460	469661	50.701	1919282	9.418
220	480449	51.865	1983459	9.733	470	461177	49.785	1892298	9.285
230	474204	51.191	1965348	9.644	480	459438	49.597	1884504	9.247
240	469884	50.725	1947259	9.555	490	465346	50.235	1895723	9.302
250	468938	50.623	1930288	9.472	500	463730	50.06	1890510	9.277

Table 5.4 Error rates for threshold .3

From the above results we can easily say that epoch 480 gives less error for the 3 considered threshold value. If we plot the result in bar chart for threshold value .3 then it will be like following

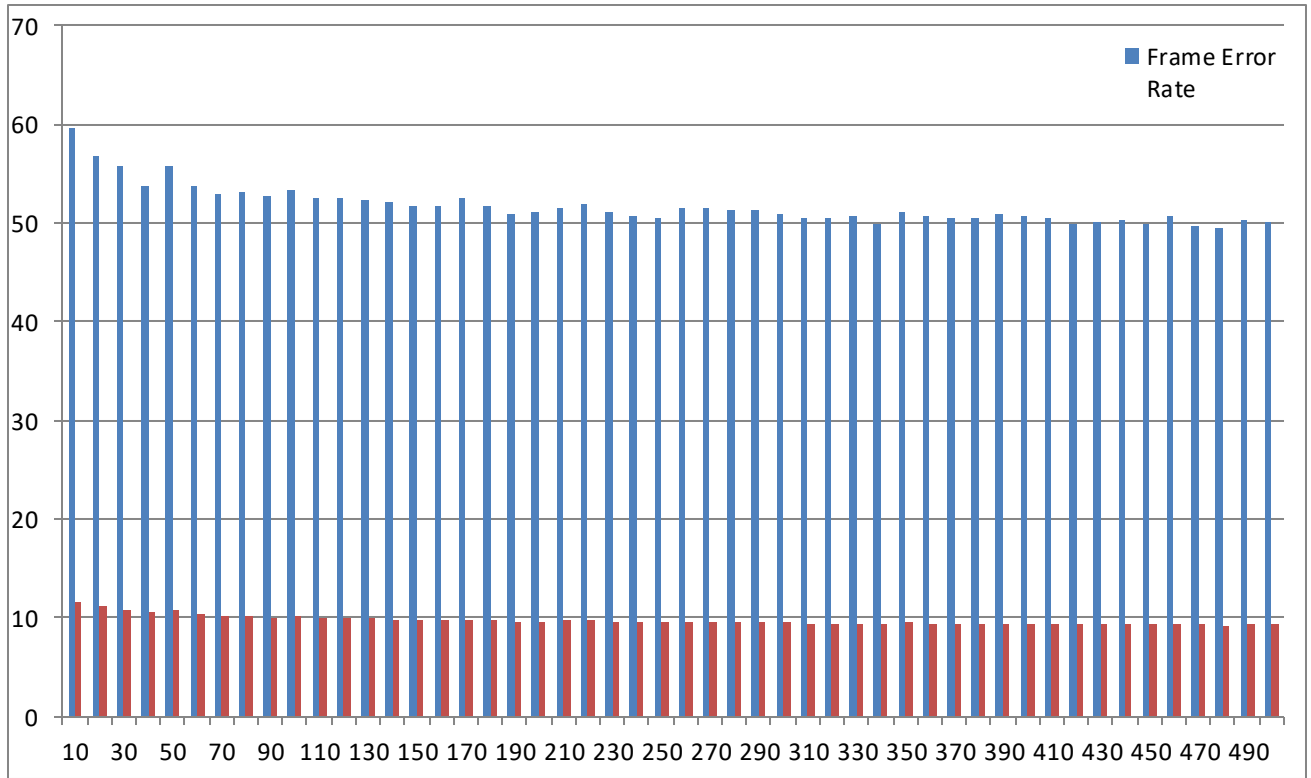


Figure 5.11 Bar chart of the error results for threshold .3

5.4 Normalization

The 480 epoch AF values generated by the MLN are log normally distributed because we have used sigmoid as the activation function. As Gaussian mixture model has been used so need to convert the lognormal distribution to normal distribution.

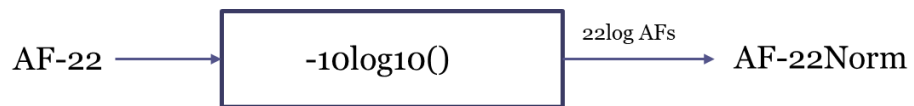


Figure 5.12 Normalizing AF values

5.5 Hidden Markov Model Training

5.5.1 Generating phone level transcriptions for training

While training HMMs, each file that has the training set must have an associated phone level transcription. A flat-start scheme will be used to bootstrap a set of models. For this, two set of phone transcriptions is needed. Initially there will be no short pause (sp) model between words. After we get appropriate phone models, an **sp** model will be embedded between the consecutive words to handle pauses introduced by the speaker.

Assuming that the file **words.mlf** stores word level MLF, a phone level transcription **phones0.mlf** will be generated by executing the command below.

```
HLEd -l '*' -d dict -i phones0.mlf mkphones0.led words.mlf
```

Figure 5.13 illustrates the process

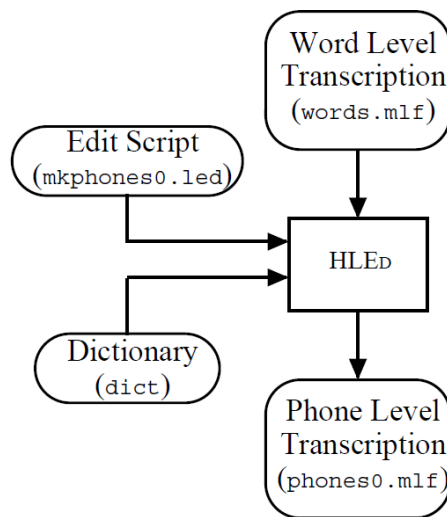


Figure 5.14 Generation of phone level transcription

5.5.2 Generation of mono phone model

5.5.2.1 Creating prototype model

The first step while training HMM is to create a prototype model. The reason of the creation is to specify the topology of the model. The file **proto_5states_39dim** contains “3-state left-right with no skips” topology which is good for phone-based systems .

Figure 6.13 shows a configuration file which specifies input output type and vector dimension.

```

HTK_39dpf.config - WordPad
. . . . . 1 . . . . . 2 . . . . . 3 . . . . . 4
SOURCEKIND = USER      # 39 dim (DPF values)
TARGETKIND = USER      # 39 dim (DPF values)

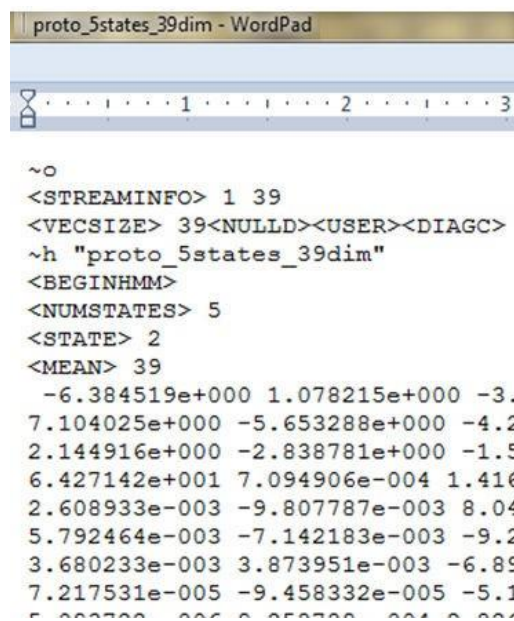
RAWENERGY = F
  
```

Figure 5.15 HTK_39dpf.config file structure

Command executed:

```
HCompV -T 1 -C HTK_39dpf.config -S Train.scp -m -v 0.01 -f 0.01 -M hmm0\  
proto_5states_39dim'
```

After execution a new edition of proto in hmm0\ directory is created where the global speech means and variances replaces the zero means and unit variances. Following figure shows the new version.



```
~0  
<STREAMINFO> 1 39  
<VECSIZE> 39<NULLD><USER><DIAGC>  
~h "proto_5states_39dim"  
<BEGINHMM>  
<NUMSTATES> 5  
<STATE> 2  
<MEAN> 39  
-6.384519e+000 1.078215e+000 -3.  
7.104025e+000 -5.653288e+000 -4.2  
2.144916e+000 -2.838781e+000 -1.5  
6.427142e+001 7.094906e-004 1.416  
2.608933e-003 -9.807787e-003 8.04  
5.792464e-003 -7.142183e-003 -9.2  
3.680233e-003 3.873951e-003 -6.89  
7.217531e-005 -9.458332e-005 -5.1  
5.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
```

Figure 5.16 Segment of updated proto_5states_39dim

5.5.2.2 Creating HMM definition

This step illustrated in Figure 6.15, basically creates an HMM definition in the directory

hmm0\model for each mono phones listed in **monophones0** file.

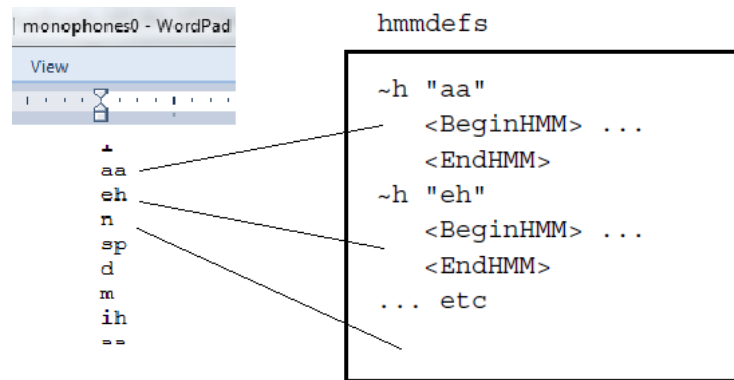


Figure 5.17 Creation of hmmdef

5.5.2.3 Re-estimation

Using the re-estimation tool HERest, `hmm0/hmmdefs` is re-estimated. The command executed is as follows.

**HERest -A -D -T 1 -C HTK_39dpf.config -I phones0.mlf -t 250.0 150.0 1000.0 -S Train.scp -H
hmm0/model/macros -H hmm0/model/hmmdefs -M hmm1 hmm0/model/monophones0**

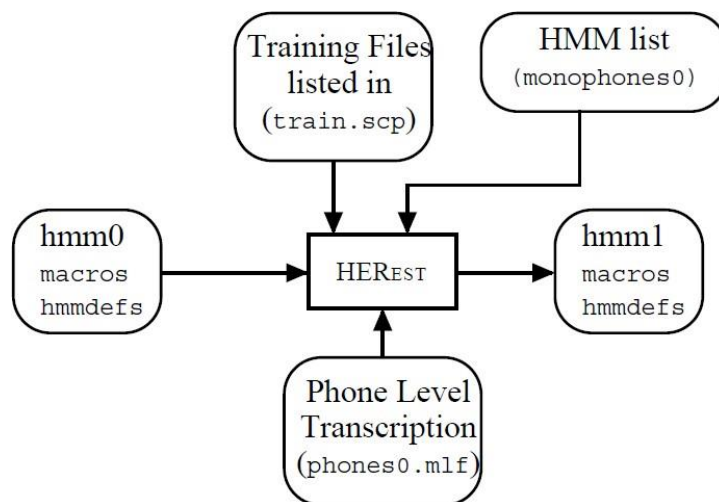


Figure 5.18 Illustration of this step

5.5.2.4 Fixing Silence Model

In the previous step a left to right 3 state Hidden Markov Model for each phone and silence model have been generated. Next, an extra link from states 2 to 4 and from 4 to 2 is added to the silence model. Figure 6.17 illustrates this step. The way of making more robust model is to allow every state to soak in a variety of noises in the training data.

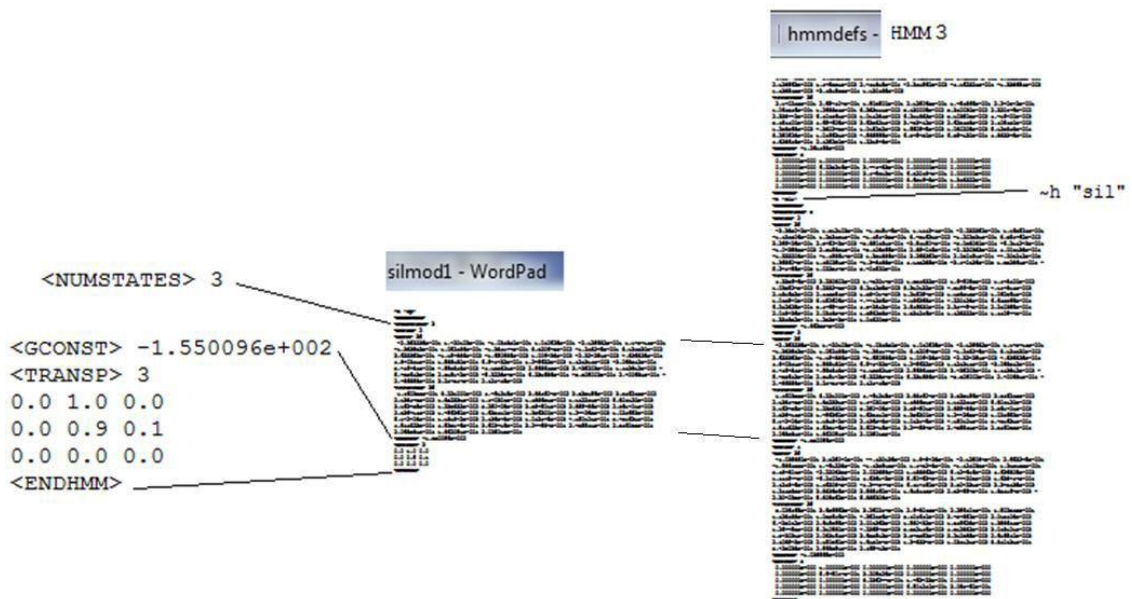


Figure 5.19 Fixing the silence model

The central state is established as a short pause model in the last of **hmm4hmmdefs** file. This is done using HHED tool. Here the file **monophones1** is exactly same as monophones0 with a **sp** added at the end. The **sil.hed** header file change transition matrix dimension (5 state **sil** to 3 state **sp**),

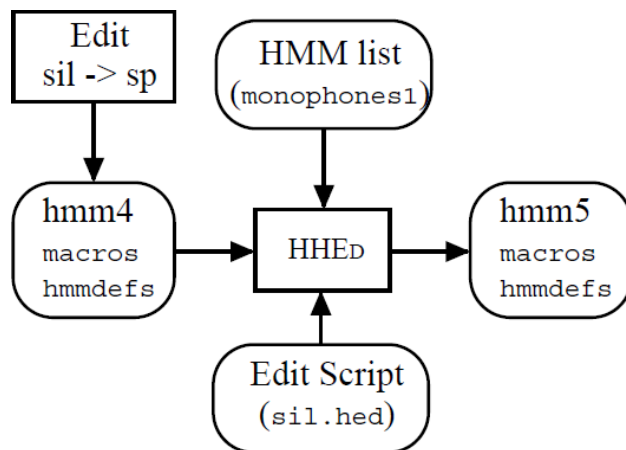


Figure 5.20 Illustration of this process

5.5.2.4 Re-estimation after fixing the silence model

This step re-estimates **hmm5\hmmdefs** using the re-estimation tool **HERest**,

Figure 6.19 illustrates this process

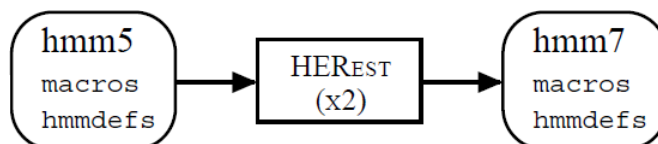


Figure 5.21 Re-estimation up to hmm7.

5.5.3 Generation of tri phone models

After getting a set of mono phone HMMs, the next step of the process is to triphone HMMs. This can be done following 2 steps. First, the mono phone transcriptions are changed into tri phone transcriptions. By copying the mono phones and then re-estimating, triphone models are created. Secondly, acoustic states similar to these tri phones are combined to make sure all state distributions estimation robustly.

5.5.3.1 MLF to LAB file conversion

This step is actually a pre-processing step, which involves slicing the words.mlf file to individual label files using mlfToLib_Generator.exe. Figure 4.19 shows a portion of mlf and lab file.

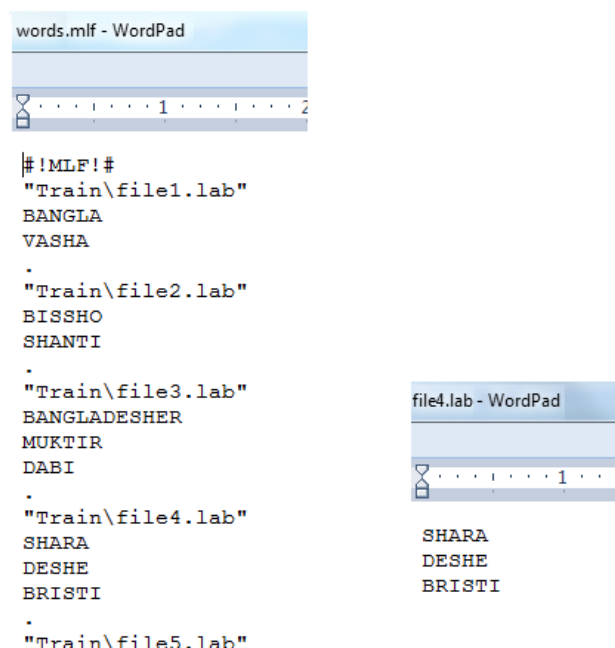


Figure 5.22 mlf to lab file generation.

5.5.3.2 Realigning the data

The training data are realigned and new transcriptions are created using the phone models. This can be done by executing following command.

```
HVite -A -D -T 1 -l * -o SWT -b SENT-END -C HTK_39dpf.config -H hmm7/macros -H  
hmm7/hmmdefs -i aligned.mlf -m -t 250.0 150.0 1000.0 -y lab -a -I words.mlf -S Train.sc  
dict monophones1> HVite_log.txt
```

After execution a file named aligned.mlf is generated, which is similar to phones1.mlf .

5.5.3.3 Re-estimation

hmm7/hmmdefs is re-estimated up to **hmm9** using the re-estimation tool **HERest**.

The command executed is as follows:

```
HERest -A -D -T 1 -C HTK_39dpf.config -I aligned.mlf -t 250.0 150.0 3000.0 -S train.sc  
-H hmm7/macros -H hmm7/hmmdefs -M hmm8 monophones1>hmm8.log
```

5.5.3.4 Generate Context dependent Triphones

Context-dependent triphones are generated by replicating the mono phones and then re-estimating them by tri phone transcriptions. This can be done using **mktri.led** script in a tool **HLEd** as shown in Figure 6.21 which converts all monophones to triphones.

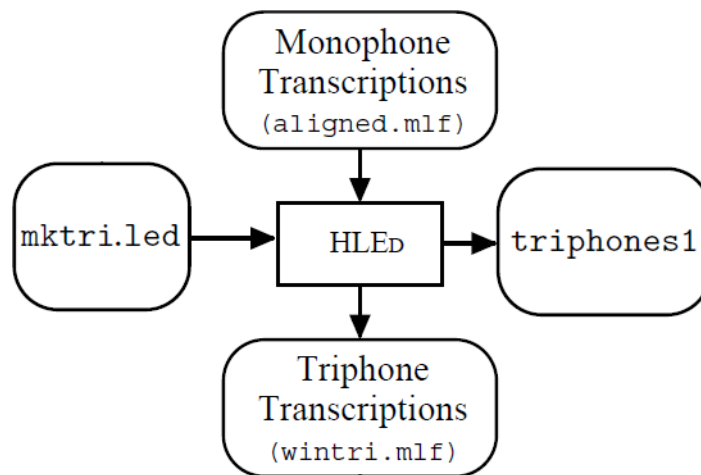


Figure 5.23 Triphone generation

The **mktri.led** script file looks like this,

WB sp

WB sil

TC

Here **sp** and **sil** are word boundary symbols and **TC** links it with another script file **mktri.hed**.

The following command is executed to achieve this is

HLEd -A -D -T 1 -n triphones1 -l * -i wintri.mlf mktri.led aligned.mlf

Figure 6.22 shows segment of wintri.mlf and triphones1 files

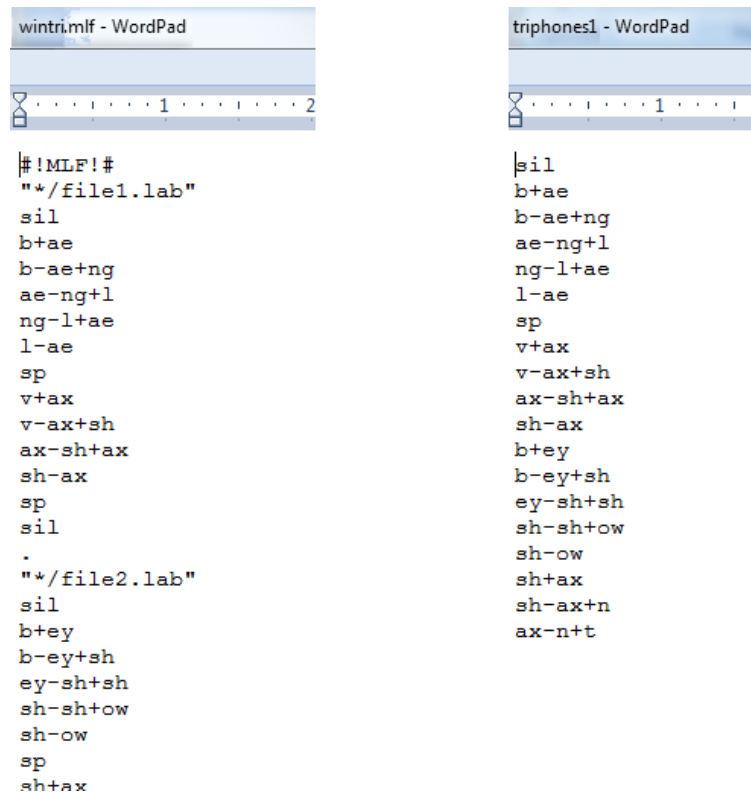


Figure 5.24 A segment of wintri.mlf and triphones1 files

Using the following perl script, another script file **mktri.hed** is generated.

```
status=dos('C:/perl/bin/perl maketrihed monophones1 triphones1')
```

The file **mktri.hed** holds CL clone command and subsequently TI command to combine the transition matrices in each triphone set.

The command CL takes the file that contains the list of tri phones (and bi phones) as its argument. For each model in a-b+c form, it searches for the mono phone and duplicate it in **hmm10\hmmdefs**. This is re-estimated further up to **hmm12** using the **HERest** tool. Figure 4.22 provides an elaborate view of this step.

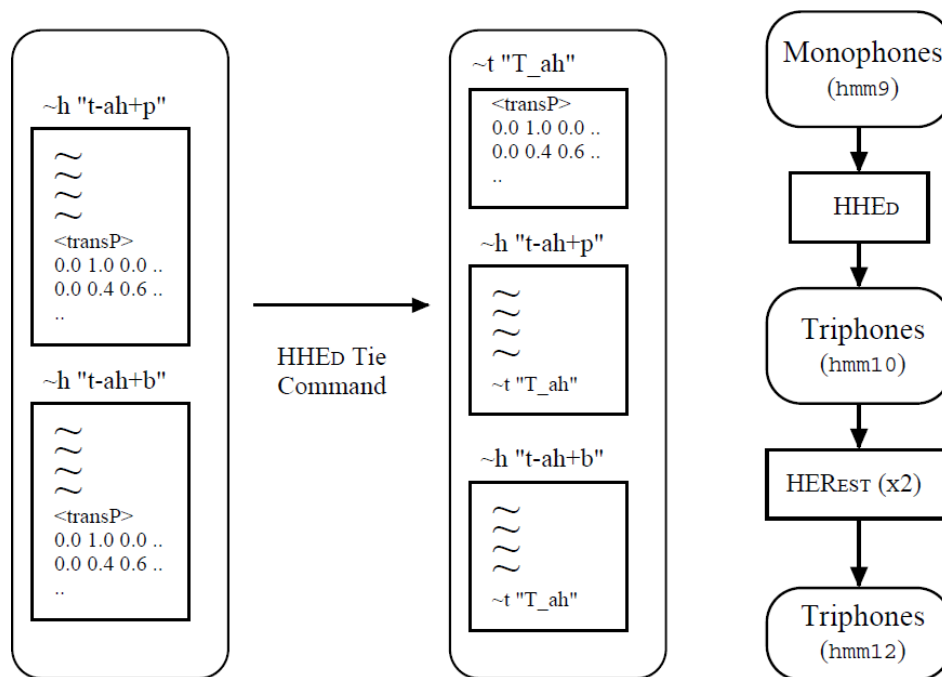


Figure 5.25 Tying transition matrices

5.5.3.5 Tied states HMM

The output of the foregoing step is a tri phone HMM set where all the tri phones are in a phone set and share the same transition matrix. While estimating these models the variances in the output distributions will be floored as there is less data corresponding with most of the states. The final step is to tie states in tri phone sets to share data and make vigorous parameter estimates.

Executing the **HDMAN** command creates 2 files, **dict-tri** and **fulllist**

HDMAN -A -D -T 1 -b sp -n fulllist -g global.ded -l flog dict-tri dict

The **global.ded** file contains the following lines.

AS sp RS cmu

MP sil sil sp

The contents of **dict-tri** and **dict** is almost similar, while the file **fulllist** contains a complete list of mono phones and tri phones used. This file is further updated by adding **sp** and **sil** model and saved as **fulllist1**.

A file named **tree.hed** is needed for tying the HMMs. By running the following command the tied state HMMs are created:

```
HHed -A -D -T 1 -H hmm12/macros -H hmm12/hmmdefs -M hmm13 tree.hed  
triphones1
```

This generates **hmm13\hmmdefs** and a file named **tiedlist**, which keeps track of all tied states. **hmm13** can be further re-estimated using **HERest** tool. Figure 4.24 shows the summary of this step.

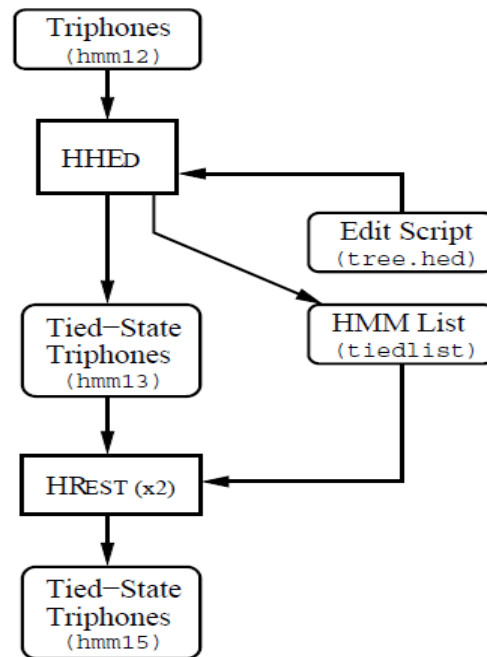


Figure 5.26 Summary of this step

The basic data training steps are up to this. Further re-estimation is possible, but hmm15 seems optimized and gave better result while testing.

5.4 Testing with test data

Now, the test data has been prepared and the recognition network with the dictionary is constructed. It's time to use the **HVite** recognizer tool and valuate the results using the **HResults** analysis tool.

Files needed for test:

Train files: hmm15\macros,hmmdefs and tiedlist

Test files: Test *.vec files, generated from the AF files

We assume that **testf.scp** contains a test file list. Every test file is going to be recognized and its transcription will be saved to **testrecout_mix1.mlf** by running the following command:

```
HVite -A -D -T 1 -H D:\Thesis\practice \Test\hmm15\macros -H D:\Thesis\practice  
\Test\hmm15\hmmdefs -C F:\experiment\Test\HTK_dpf.config -S D:\Thesis\practice  
\Test\Testf.scp -l D:\Thesis\practice \Test\words\ -i D:\Thesis\practice  
\Test\testrecout_mix1.mlf -w D:\Thesis\practice \Test\testwdnet -p 0.0 -s 5.0  
D:\Thesis\practice \Test\voxforge_lexicon D:\Thesis\practice \Test\tiedlist>  
D:\Thesis\practice \Test\test_mix1.log
```

Assuming that word level transcriptions for all test file is in the MLF **words_Test.mlf**, the performance can be evaluated by executing the command as follows:

```
HResults -I F:\experiment\Test\words_Test.mlf F:\experiment\Test\tiedlist  
F:\experiment\Test\testrecout_mix1.mlf>resultmix1.txt
```

The statistical result has been stored in the file **resultmix1.txt**. This step is repeated for mix 2,4 and 8. The whole step is summed up in Figure 4.25

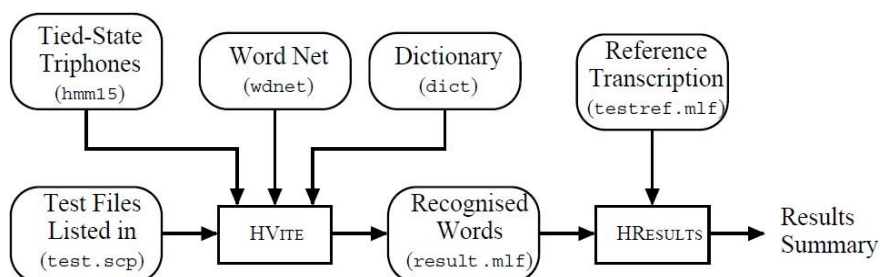


Figure 5.27 Summary of testing step

Chapter 6 Experiments and Results

6.1 Performance figures

We'll measure the performance in 3 category. These are:

1. Word Correct Rate (WCR) :

It calculates the amount of words correctly recognized from all the words. This can be written as follows

$$\%wordCorrectRate = \frac{N - S - D}{N} \times 100\%$$

Here, N= Total No of Words
D= No of Deletion
S=No of Substitution

2. Word Accuracy (WA) :

For Accuracy calculation it considers no of inserted words. The equation for word accuracy calculation is:

$$\%wordAccuracy = \frac{N - S - D - I}{N} \times 100\%$$

Here, I=No of Insertion

3. Sentence Correct Rate (SCR) :

For calculating sentence correct rate, sentences that are recognized correctly from total no of sentences are considered. The equation can be written as

$$\%sentenceCorrectRate = \frac{H}{N} \times 100\%$$

Here, N=Total No of Sentences

H=Total No of Correctly Recognized Sentence

6.2 Experimental Setup

The frame length is set 25 ms and the frame rate is set 10 ms to get MFCCs from input speech. MFCC features are 39 dimensional and AFs are 22 dimensional.

For constructing an accurate continuous word recognizer word accuracy (WA), sentence correct rate (SCR) and word correct rate (WCR) and for test corpus D2 are calculated using the built classifier. This HMMs with 5 states, 3 loops, and left-to-right models are designed using the train corpus D1. The output probabilities in the HMMs are represented in the Gaussian mixture form and diagonal matrices are used. The mixture components are set to one, two, four and eight.

For the evaluation of the performance of the proposed system we have constructed these experiments:

(a) MFCC-39 [Train: D1 dataset, Test: D2 dataset]

(b) AF-22 [Train: D1 dataset, Test: D2 dataset]

6.3 Experimental Results and Discussion

Mixture Component	Methods	Sentence Recognition Performance	
		Total sentence=1000	
		Correctly Recognized Sentences, H	Incorrectly Recognized Sentences, S
Mix 1	MFCC-39	909	91
	AF-22	971	29
Mix 2	MFCC-39	909	91
	AF-22	963	37
Mix 4	MFCC-39	16	984
	AF-22	955	45
Mix 8	MFCC-39	926	74
	AF-22	964	36

Table 6.1 Sentence Recognition Performance

Table 6.1 shows sentence correct rates for both MFCC-39 [Standard method] and AF-22 [Proposed method] where total no of sentences are 1000. For mixture component 1 maximum correctly recognized sentences is 971 among 1000 sentences for our proposed method which means minimum error in sentence recognition performance. For mixture component 4, standard MFCC-39 method performs worst where correctly recognized sentences are 16 among 1000 sentences. But this result improves in mixture component 8.

Mixture Component	Methods	Word Recognition Performance (Total Words=3290)			
		Correctly Recognized Words, H	Deletion, D	Substitution, S	Insertion, I
Mix 1	MFCC-39	3017	64	209	16
	AF-22	3207	21	62	6
Mix 2	MFCC-39	3017	64	209	16
	AF-22	3179	30	81	6
Mix 4	MFCC-39	122	653	2515	530
	AF-22	3160	36	94	7
Mix 8	MFCC-39	3076	53	161	11
	AF-22	3186	23	81	9

Table 6.2 Word Recognition Performance

Table 6.2 shows word correct rates for both the methods. There are total 3290 words in 1000 sentences. Our proposed method performs best in mixture component 1 where total 3207 words are correctly recognized out of 3290 words. Only 21 words have been deleted. In mixture component 4, total 2515 words have been substituted in the MFCC method and only 122 words are correctly recognized. Maximum number of deletion (653) also occurs in this method for mixture component 4. Minimum number of insertion occurs in mixture component 1 and 2 for AF-22 method which is 6.

Mixture Component	Methods	Word Correct rate(%)	Word Accuracy(%)	Sentence Correct rate(%)
Mix 1	MFCC-39	91.70	91.22	90.90
	AF-22	97.48	97.29	97.10
Mix 2	MFCC-39	91.70	91.22	90.90
	AF-22	96.63	96.44	96.30
Mix 3	MFCC-39	3.71	12.40	1.60
	AF-22	96.05	95.84	95.50
Mix 4	MFCC-39	93.50	93.16	92.60
	AF-22	96.84	96.57	96.40

Table 6.3 Recognition Performance (%)

From table 6.3, it is seen that our proposed method provides the highest word accuracy of 97.29% and highest sentence correct rate of 97.10% for mixture component 1.

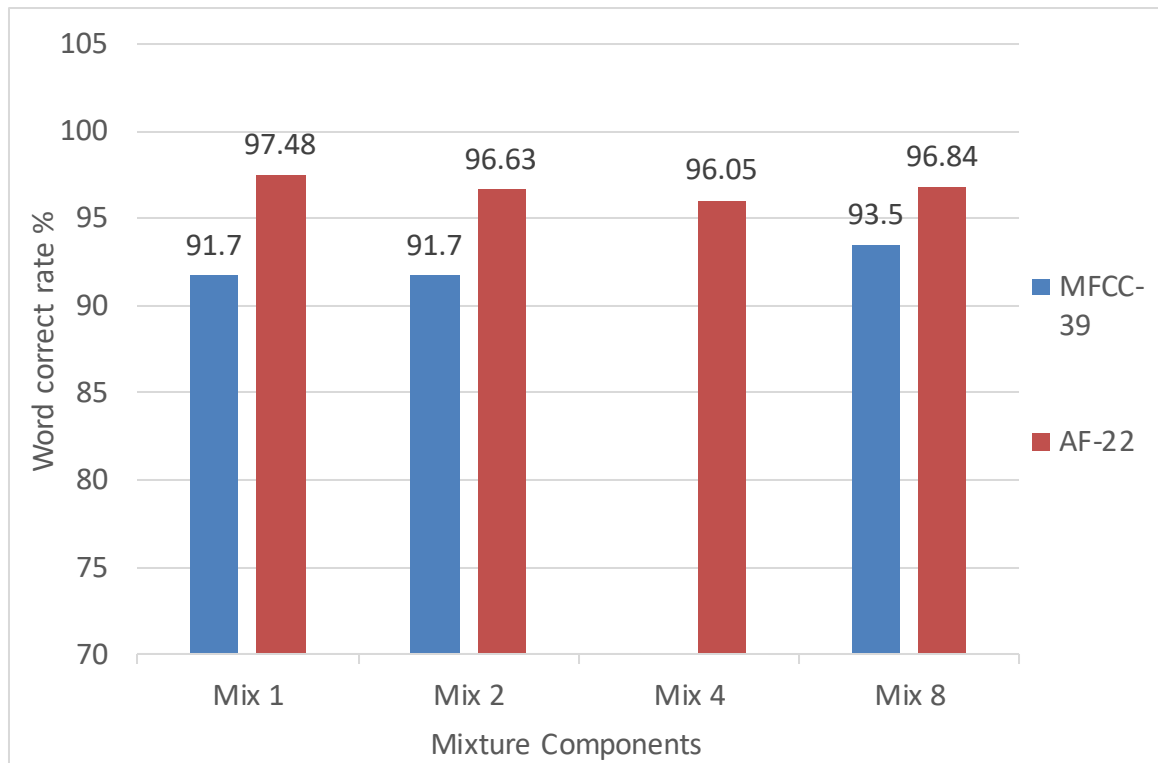


Figure 6.1 Word Correct rate comparison between MFCC-39 and AF-22

From Figure 6.1, it is clearly seen that proposed method's bar is higher than the existing standard method and the Bar labels are 97.48%, 96.63%, 96.05%, 96.84% for mixture components 1,2,4 and 8 respectively.

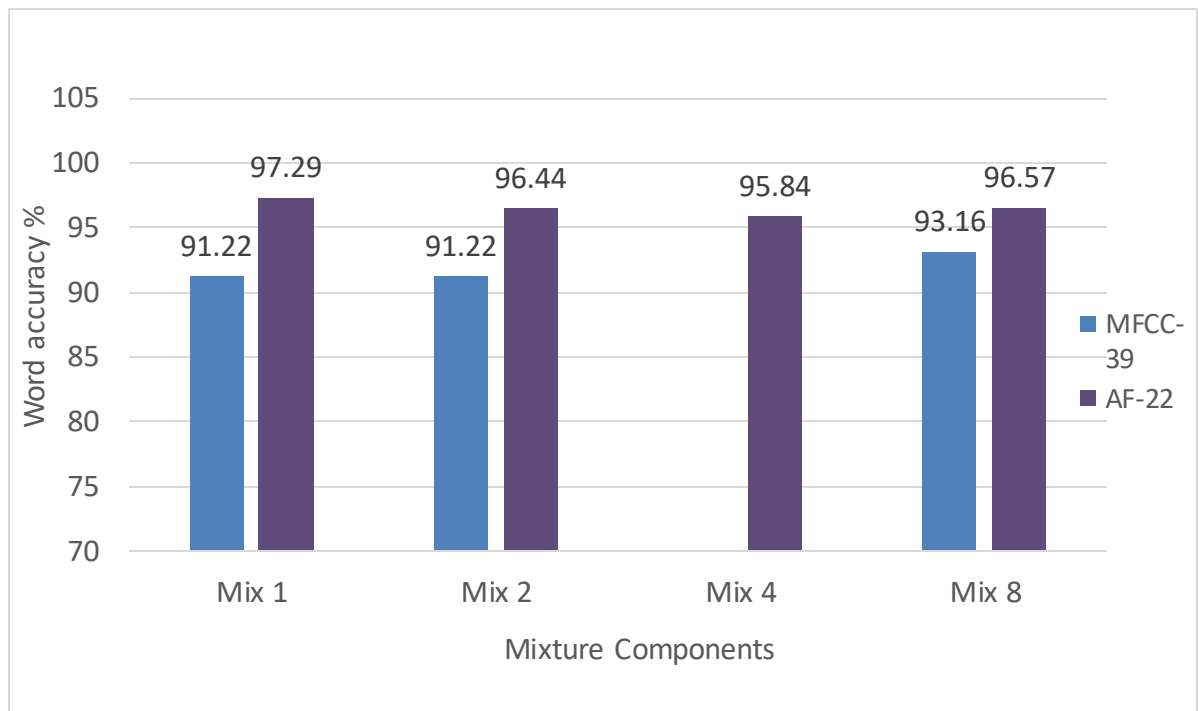


Figure 6.2 Word Accuracy comparison between MFCC-39 and AF-22

From Figure 8, we can see that proposed method's bar for word accuracy is higher than the existing standard method and the Bar labels are 97.29%, 96.44%, 95.84%, 96.57% for mixture component 1,2,4 and 8 respectively.

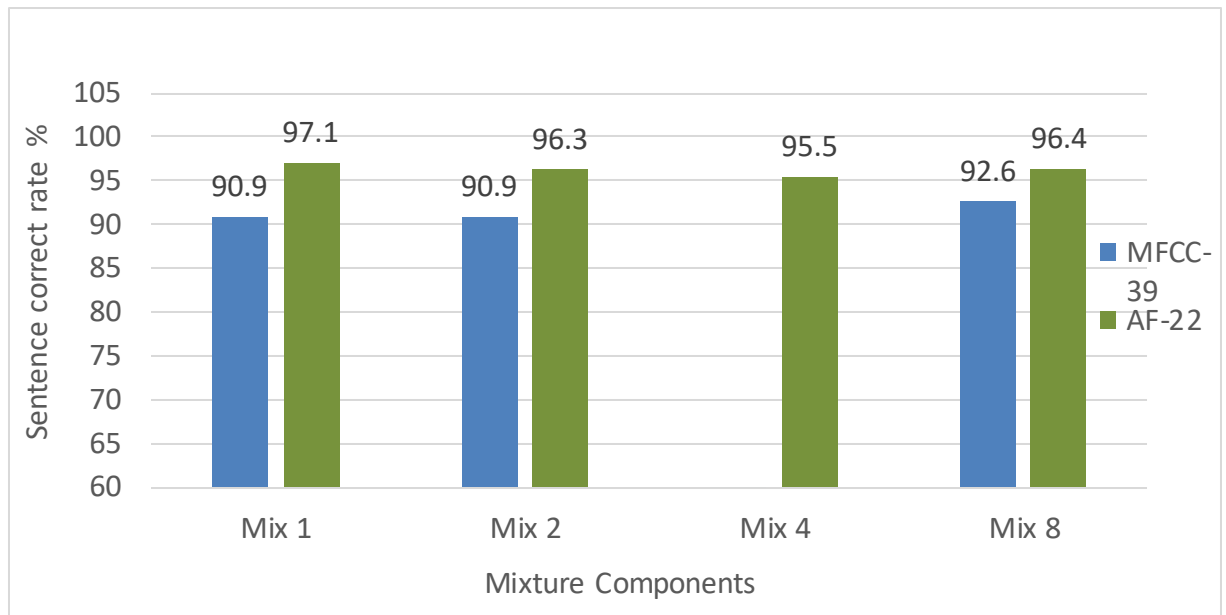


Figure 6.3 Sentence correct rate comparison between MFCC-39 and AF-22

From Figure 6.3, we can see that sentence correct rate for proposed method is higher than the existing standard method and the Bar labels are 97.29%, 96.44%, 95.84%, 96.57% for mixture component 1,2,4 and 8 respectively.

Chapter 7 Conclusion and Future works

7.1 Conclusion

The main concentration of this research was to improve the performance of the existing standard ASR system. The report covered the basics of speech recognition. The following conclusions can be drawn from the experiments:

1. The AF-22 based system provides huge improvement on Bangla word recognition accuracy for test data.
2. A higher rate of Bangla word correction for test data is also obtained by the AF-22 based system.
3. Sentence correct rate also has been improved.
4. We have learned how we can integrate neural network into the recognition process.

7.2 Future Work

Speaker independency is a major fact in many experimental applications. The methods introduced in this report 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 very few researches for Bangla automatic speech recognition system exists, there is a need to conduct lot more research works to enrich this system. So, our research based on articulatory feature using neural network in the area of Bangla Automatic Speech Recognition is a good approach to achieve better word accuracy, word correct rate and sentence correct rate. In future we would like to work on

- Gender Independent ASR system
- Incorporate RNN based system
- Bangla phoneme recognition system.

References

- [1] M. J. Alam, N. Uzzaman, and M. Khan. N-gram based statistical grammar checker for bangla and english. Proc. of 9th International Conference on Computer and Information Technology (ICCIT 2006), 2006.
- [2] L. R. Bahl, P. F. Brown, P. V. De Souza, and R. L. Mercer. Acoustic markov models used in the tangora speech recognition system. In Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, 1988.
- [3] K. H. Davies, R. Biddulph, and S. Balashek. Automatic speech recognition of spoken digits. J. Acoust. Soc. Am., 1952.
- [4] L. Deng, M. Lennig, V. N. Gupta, and P. Mermelstein. Modeling acousticphonetic detail in an hmm-based large vocabulary speech recognizer. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, 1988.
- [5] A. M. Derouault. Context-dependent phonetic markov models for large vocabulary speech recognition. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, 1987.
- [6] N. R. Dixon and H. F. Silverman. The 1976 modular acoustic processor (map). IEEE Trans. Acoustics, Speech and Signal Processing, 1977.
- [7] S. Dupont, C. Ris, L. Couvreur, and J. Boite. A study of implicit and explicit modeling of coarticulation and pronunciation variation. In Proc. of InterSpeech, Lisbon, 2005.
- [8] J. Ming et. al. Improved phone recognition using bayesian triphone models. In Proc. ICASSP, 1998.
- [9] X. D. Huang et. al. Hidden markov models for speech recognition. Edinburgh: Edinburgh University Press, 1990.
- [10] M. A. Hasnat, J. Mowla, and M. Khan. Isolated and continuous bangla speech recognition: Implementation performance and application perspective. In Proc. International Symposium on Natural Language Processing (SNLP), 2007.

- [11] S. A. Hossain, M. L. Rahman, and F. Ahmed. Bangla vowel characterization based on analysis by synthesis. Proc. WASET, 2007.
- [12] S. A. Hossain, M. L. Rahman, F. Ahmed, and M. Dewan. Bangla speech synthesis, analysis, and recognition: an overview. In Proc. NCCPB, Dhaka, 2004.
- [13] A. K. M. M. Houque. Bengali segmented speech recognition system. Undergraduate thesis, BRAC University, Bangladesh, 2006.
- [14] D. Jurafsky, W. Ward, Z. Jianping, K. Herold, Y. Xiuyang, and Z. Sen. What kind of pronunciation variation is hard for triphones to model? In Proc. of ICASSP, 2001.
- [15] D. Jurafsky, W. Ward, Z. Jianping, K. Herold, Y. Xiuyang, and Z. Sen. What kind of pronunciation variation is hard for triphones to model? In Proc. of ICASSP, 2001.
- [16] 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 (IC- CIT02), 2002.
- [17] S. P. Kishore, A. W. Black, R. Kumar, and R. Sangal. Experiments with unit selection speech databases for indian languages. Carnegie Mellon University.
- [18] K.F. Lee. Context-dependent phonetic hidden markov models for speaker-independent continuous speech recognition. IEEE Trans. Acoust. Speech Signal Processing, 1990.
- [19] J. Matousek, Z. Hanzlcek, and D. Tihelka. Hybrid syllable/triphone speech synthesis. In Proc. of InterSpeech, Lisbon, 2005.
- [20] T. Nitta. Feature extraction for speech recognition based on orthogonal acoustic-feature planes and kda. In Proc ICASSP, 1999.
- [21] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. Proc. IEEE, 1989.
- [22] K. J. Rahman, M. A. Hossain, D. Das, T. Islam, and M. G. Ali. Continuous bangla speech recognition system. In Proc. 6th International Conference on Computer and Information Technology (ICCIT03), 2003.
- [23] R. M. Schwartz, Y. Chow, O. Kimball, S. Roucos, M. Krasner, and

- J. Makhoul. Context-dependent modeling for acoustic-phonetic recognition of continuous speech. In Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, 1985.
- [24] R. M. Schwartz, J. Klovstad, J. Makhoul, and J. Sorensen. A preliminary design of a phonetic vocoder based on a diphone model. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, 1980.
- [25] R. Thangarajan, A. M. Natarajan, and M. Selvam. Word and triphone based approaches in continuous speech recognition for tamil language. WSEAS Transactions on Signal Processing, 2008.
- [26] S. J. Young, J. J. Odell, and P. C. Woodland. Tree-based state tying for high accuracy acoustic modelling. In Proc. of ARPA Workshop on Human Language Technology, 1994