Bangla Automatic Phoneme Recognition

Shrobana Barai Shweta
Student Id: 011131132
Irine Parvin
Student Id: 011132095
Nasrin Jahan Liza
Student Id: 011132016
Aziza Tun Saida
Student Id: 011131052

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

United International University
Dhaka, Bangladesh
May 2018

Declaration

We, Shrobana Barai Shweta, Irine Parvin, Nasrin Jahan Liza and Aziza Tun Saida declare that this thesis titled “**Bangla Automatic Phoneme Recognition**” presented in it is our own. We confirm that:

- This work was done wholly or mainly while in candidature for a BSc degree at United International University.
- Where any part of this thesis has previously been 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 my own work.
- We have acknowledged all main sources of help.
- Where the thesis is based on work done by ourselves jointly with others, we have made clear exactly what was done by others and what we have contributed ourselves.

____

Name: Shrobana Barai Shweta
ID: 011131132
CSE

____

Name: Irine Parvin
ID: 011132095
CSE

____

Name: Nasrin Jahan Liza
ID: 011132016
CSE

____

Name: Aziza Tun Saida
ID: 011131052
CSE

Certificate

I do hereby declare that the research works embodied in this thesis entitled "**Bangla Automatic Phoneme Recognition**" is the outcome of an original work carried out by Shrobana Barai Shweta, Irine Parvin, Nasrin Jahan Liza, Aziza Tun Saida 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 & MSCSE Coordinator
United International University, Dhaka, Bangladesh.

Abstract

Speech recognition is now known a most challenging problem. People are trying to use speech recognition in so many applications to make their work easier. In Bengali language there are so many varieties in pronunciation. They talk in their regional language.

In our thesis work we have tried to work with these different languages of Bangla. We took their voice recordings and made experiments.

We have tried to convert the recordings into text. To recognize the device with the language we tried our level best. Here we used MFCC inputted to the HMM based classifier for getting Phoneme recognition performance.

Acknowledgement

We are very grateful to Almighty Allah that we have completed our thesis successfully. We would like to express our sincere gratitude to our honorable supervisor Dr. Mohammad Nurul Huda, Professor & MSCSE coordinator, United International University (UIU), Dhaka, Bangladesh. He continuously supported us in our every work and gave his valuable time, encouragement, guidance and motivation.

Besides our advisor we would like to thank the department of Computer Science and Engineering (CSE) of UIU for giving us the chance to do the thesis work.

Finally, we are thankful to our parents for their encouragement, support, motivation, suggestions and infinite love.

Table of Contents

List of Figures:

List of Tables:

Chapter 1

Introduction

1.1. Introduction

To exchange information between two human beings speaking and listening is the most natural way, as we learn it by born? A machine knows nothing about human voice. To make a machine familiar with human's voice people have tried for centuries to build a machine.

1.2. Voice recognition

Human voice can only understand by other human. Now a day's people make a machine which can decode the human voice and it is done by computer software program or hardware device. The process of decoding human's voice by a machine is called "voice recognition". Without using any kind of mouse or keyboard or press any button it is used to write or operate any device or perform commands. For example, if a person says 'welcome' then speech recognizer will convert it into a text on the device.

1.3. Background

(i) Many phonemes are in Bangla spoken language.
(ii) For Phonetic typewriter Phoneme Recognizer is required.
(iii) For Speech Therapy Phoneme Recognizer is needed.

1.4. Importance

In recent years voice recognition has become very popular. People are using this very often for their personal use or in many other organizations. It provides various benefits and it is widely used.

(i) Speech recognition is faster than typing: When people talk it takes less time to generate the speech. And when they write it takes so many times and require some purpose. Though typing is time consuming that's why speech recognition is faster.

(ii) Battle Field Simulation: Suppose in the battle field soldiers are in critical situation. They have no time to text anyone. Within that short period of time they need to deliver some command as soon as possible. That time they can use voice recognition to make it effortlessly.

(iii) Email editor by voice: When a person types something he can make so many mistakes and it reduces his efficiency. To use the time properly and make him efficient in every part of life he can use speech recognition. It turns him into a typist who has the high speed.

1.5. Objective

In recent time around the world voice recognition is widely researched topic. To do works on speech recognition many scientists and researchers are remain busy**.** English is the mostly used language on speech recognition. And many other languages in the world also have used speech recognizer of its own. But Bengali Language is not so enriched in a speech recognizer. Bengali speech recognizer is only researched by very few people. But in our thesis we have a little try to build a train Bangla speech recognizer and we hope it will enrich our mother tongue. Our aim is to construct a Bangla Phoneme Recognizer using Acoustic Feature, MFCC (Mel Frequency Cepstral Coefficient).

Chapter 2

2.1 Phoneme Recognition

Phoneme is a unit of sound in speech that can distinguish one word from another. Like as: pat, pad, bat, bad etc.

To do phoneme recognition people would like to use Sphinx. Rather than convert them into words people like to convert them into a stream of phonemes. But its result may disappoint everyone because to guide the search algorithm automatic speech recognition relies on contextual limitations.

In word decoding phoneme recognition is much less bound. That is why the error rate is high. This process is quite slow. For so many tasks like pronunciation modeling, speaker identification and voice conversion phoneme decoding plays as a helpful one.

2.2. Bangla phonetic

Generally phonemes are written in International Phonetic Alphabet (IPA). Between Bangla and English phoneme numbers are not same. But both can be expressed in IPA which can simplify comparison between the two. English has 36 phonemes, among them there are 12 vowels and 24 consonants. On the contrary Bangla has 37 phonemes, among them there are 7 vowels and 30 consonants. These vowels and consonants are shown in Table 1 and 2.

Table 1: English Phonemes

Vowels	Consonants
I i: Ʊ u: ɒ Ɔ: ʌ ɑ: ə ɜ: e æ	p b t d θ ð k ɡ f v s z ʃ ʒ tʃ dʒ n m ŋ r l w j h

Table 2: Bangla Phonemes

Vowels	*Consonants*
ɑ i e o u ɔ æ	p p^h b b^h t t^h d d^h t̪ t̪^{h} d̪ d̪^{h} c c^h ɟ ɟ^h k k^h g g^h s ʃ m n ŋ l r h w j

There are so many differences between Bangla and English IPA table. Some are shown below:

Table 3: Bangla Vowels

	Phoneme	Bangla word	Meaning
1	/ɑ/	k**a**n	ear
2	/i/	m**i**l	similarity
3	/e/	p**e**t	belly
4	/o/	g**o**l	round
5	/u/	b**oo**k	chest
6	/ɔ/	b**o**k	stork
7	/ æ/	b**a**ng	frog

Table 4: English Vowels

	phoneme	example
1	/t/	Ship
2	/i:/	Sheep
3	/ʊ/	Full
4	/u/	Fool
5	/ɒ/	Cot
6	/ɔ/	Caught
7	/ʌ/	Cut
8	/ɑ:/	Cart
9	/ə/	But
10	/ɜ:/	Bird
11	/e/	Pen
12	/æ/	Man

Table 5: Bangla Consonants

	Phoneme	*Bangla word*	*Meaning*
1.	/p/	**p**ul	bridge
2.	/p^h/	**ph**ul	flower
3.	/b/	**b**or	bridegroom
4.	/b^h/	**bh**or	weight
5.	/t/	**t**ok	sour
6.	/t^h/	**th**ok	cheat
7.	/d/	**d**al	pulse
8.	/d^h/	**dh**al	shield/slope
9.	/t̪/	**t**al	rhythm
10.	/t̪^{h}/	**th**al	plate
11.	/d̪/	**d**an	donation
12.	/d̪^{h}/	**dh**an	paddy
13.	/c/	**ch**al	rice/conspiracy/ roof
14.	/c^h/	**chh**al	tree bark
15.	/ɟ/	**j**al	net/fake
16.	/ɟ^h/	**jh**al	hot tasting
17.	/k/	**k**al	yesterday
18.	/k^h/	**kh**al	canal

Table 6: English Consonants

	phoneme	example		phoneme	example
1	/p/	pit	13	/b/	Bit
2	/t/	tin	14	/d/	Din
3	/k/	cut	15	/ ɡ /	Gut
4	/tʃ/	cheap	16	/dʒ/	jeep
5	/m/	map	17	/n/	Nap
6	/f/	fat	18	/v/	Vat
7	/θ/	thin	19	/ð/	then
8	/s/	sap	20	/z/	Zap
9	/ʃ /	she	21	/ʒ/	measure
10	/η/	bang	22	/h/	ham
11	/r/	run	23	/l/	left
12	/j/	yes	24	/w/	we

Here we can see that in a single word how phoneme is used. For a single phoneme a word can be changed so that we can see the difference between many words. Some peoples pronunciation is differ from one another. By applying phoneme on words can know about the exact word what people want to say.

Table 7: Bangla Vowels IPA Table

Letter	IPA	Our Symbol
অ	$/\supset /and/o/$	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 8: Bangla Consonants IPA Table

Letter	IPA	Our Symbol
ক	/k/	k
খ	$/k^h/$	kh
গ	/g/	g
ঘ	$/g^h/$	gh
ঙ	$/\eta/$	ng
চ	$/t\int/$	ch
ছ	$/t\int^h/$	chh

জ	$/d\Im/$	j
ঝ	$/d\Im^h/$	jh
ট	/t/	ta
ঠ	$/t^h/$	th
ড	/d/	da
ঢ	$/d^h/$	dha
ণ	/n/	N
ত	$/\underline{t}/$	t
থ	$/t^h/$	th
দ	$/\underline{d}$	D
ধ	$/\underline{d}^h/$	Dh
ন	/n/	N
প	/p/	p

ফ	/p^h/	ph
ব	/b/	B
ভ	/b^h/	Bh
ম	/m/	M
য	/dʒ/	J
র	/r/	R
ল	/l/	L
শ	/ʃ//s/	S
ষ	/ʃ/	S
স	/ʃ/ /s/	S
হ	/h/	H
ড়	/h̲/	Rh
ঢ়	/r̲/	Rh
য়	/e̲/	Y

Chapter 3

Hidden Markov Models

3.1. Introduction

To model sequential data Hidden Markov Model is a comparatively simple way. HMM is known as a Statistical Markov Model. It is being modeled assumed to be a Markov Process with unsighted model. As the simplest Dynamic Bayesian Network the Hidden Markov Model can be described. In speech recognition HMMs is mostly used because speech signal can be viewed as a short-time stationary signal. In Hidden Markov Model the state is not directly visible but the output which is dependent on the state is visible.

3.2. Hidden Markov Models

Recently HMMs is very popular because training them is very simple and automatic. HMMs is computationally possible to use. The HMM gives the output as a sequence of n-dimensional real-valued vector with n being a small integer such as 10 and outputting one of this in every 10 milliseconds. The vectors would consist of Mel Frequency Cepstral Coefficients (MFCCs), which are obtained by taking a Fourier transform of a short time window of speech. And at next it de-correlating the spectrum using a discrete cosine transforms (DCT), and then the first (most significant) coefficients will be taken.

Now we explain the application of Hidden Markov Models in the field of Automatic Speech Recognition. At First step in this process is to extract of spectral feature for each time frame. Now for each two consecutive time frames, t and (t+1),the hmm-based recognizer is assumed to transition from state i to state j with probability aij , or it stay in the state i with probability aii and emit an observation symbol Ot with probability density bj(Ot).

The statistical model names Hidden Markov Model uses a finite number of states. It has been used widely to model fundamental speech units in speech recognition. Because the HMM can adequately characterize both the temporal and spectral varying nature of the speech signal. Many variants exist; the simplest sub word model is a left-to-right HMM with only self and forward transitions. Within each state of the model there is an observation density function which specifies the probability of a spectral vector.

This observation density can be a discrete density, or a continuous mixture density, or called semi-continuous density or a tied-mixture density which is a set of common continuous densities whose weight determined to the model state. Tying can also be done at the HMM state level. Some of the most often used acoustic modeling approaches include:

- Maximum Likelihood (ML) Estimation of HMM.
- Maximum Mutual Information (MMI) Estimation of HMM.
- Maximum A Posteriori (MAP) Estimation of HMM.
- Minimum Classification Error (MCE) Estimation of HMM and ANN.

(i)Maximum Likelihood (ML) Estimation of HMM: Generally, Estimation of HMM parameters are achieved in a batch mode. This is done by using ML (Maximum Likelihood) approach. This ML approach is based on EM (estimation-maximization) algorithm. Besides, Segmental ML approaches have been used in a large scale. ML estimation has good asymptotic properties but the problem is, it needs a big training set to get dependable parameter estimation. To control problems related to this, some smoothing techniques, for example "Deleted interpolation", "Bayesian smoothing", may be used.

(ii)Maximum Mutual Information (MMI) Estimation of HMM: The MMI estimation procedure gives more importance to maintain the mutual information between the given acoustic data and its transcription rather than giving priority to sustain the maximum similarity between the given acoustic data and the transcription. ML estimation uses only class-specific data whereas MMI estimation considers information from data in other classes.

(iii)Maximum a Posteriori (MAP) Estimation of HMM: Maybe the best way to train sub word units is to use them to the speaking context. Through Bayesian learning, adaptive training, in a way, can be accomplished. It can be attained by maximum a posteriori estimation of HMM parameters.

(iv)Minimum Classification Error (MCE) Estimation of HMM and ANN: There is a new dimension seen in speech recognition research, that is, to design a recognizer that will be able to minimize the error rate on task-specific training data. There is a problem in this system. The problem is, the error probability is not easy to express in a close functional form. To solve this, we need an alternative way that can minimize the recognition error. For HMM-based recognizers, a family of generalized probabilistic descent (GPD) algorithms has been successfully applied. It has been done so to estimate model parameters based on the minimum classification error (MCE) criterion. The symbol o_t is a 39-dimensional feature vector with real values. Each phoneme is typically modeled using anywhere between 3 and 12 HMM states, depending on the recognizer implementation. We will give an example of a very simple HMM in Fig 1.

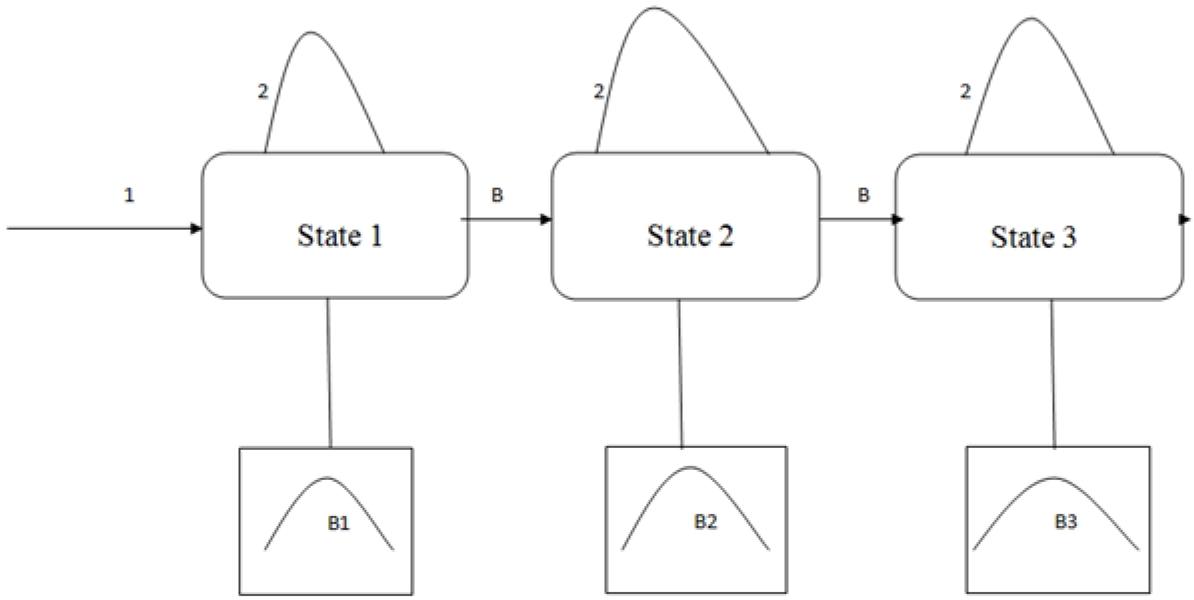

Figure 1: An HMM Example State Diagram

Chapter 4

4.1. Phoneme recognition

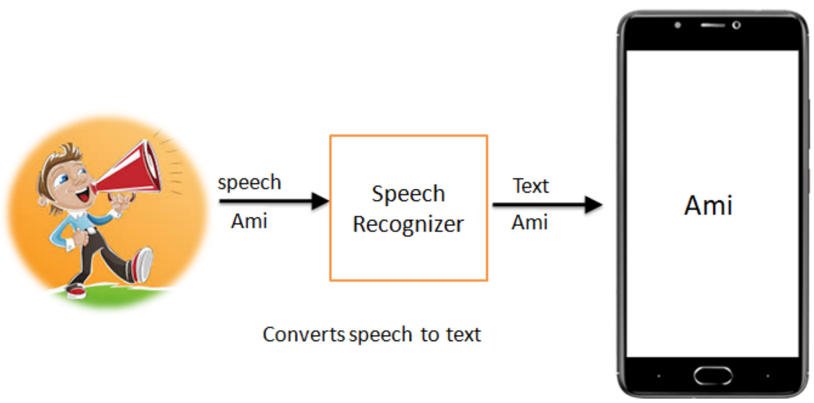

Figure 2: Convert Speech into Text

Here the speaker speaks "Ami" and the speech Recognizer converts it into text format "Ami" which is shown in the device.

4.2. Proposed System

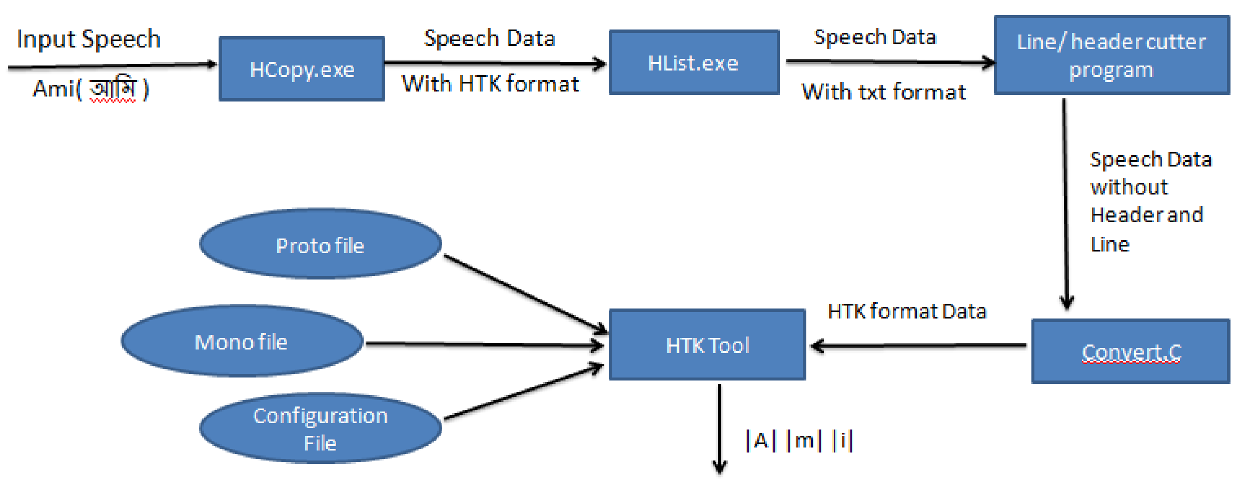

Figure 3: System Diagram

In this diagram we see that at First we take the input of speech file data or recorded file data. Then we convert all data to the HTK format by running HCopy.exe command. And then run

HList.exe command for getting all the data at .txt format. Then we run Line/Header cutter program and get all speech data without header and line. Then we run convert.C program. We get the HTK format Tool and with proto file, mono file and the configuration file we get the |A| |m| |i| as the plain text.

Chapter 5

Experiment set up and results

5.1. Introduction

All spoken languages in the world have enriched with their own literature. In the world for almost all the major spoken languages there have been many literatures in ASR used but unfortunately our Bengali language and Bengali literature is not so much enriched in ASR system. Bangla is spoken by more than 220 million people as their native language. The lack of proper speech corpus is the major problem in Bangla ASR. To build a Bangla text to speech system some efforts are taken to develop Bangla speech corpus. In our thesis paper we tried to build a ASR system for Bangla phoneme. And for this reason we first use the medium size of speech corpus of different comprises of native speakers covering almost all the major cities of Bangladesh. Then we extract the Mel Frequency Cepstral Coefficients (MFCCs) from the input speech and finally extracted features are inserted into the Hidden Markov Model (HMM) based classifier for obtaining the phoneme recognition performance.

5.2. Bangla Speech Corpus

In our thesis paper we use two types of corpuses.
(i) Training corpus.
(ii) Testing corpus.

(i)Training corpus: By 30 female speakers 3000 Bengali sentences were uttered. 100 sentences were uttered by each speaker. So this (30*100=3000) sentences are uttered for Training corpus.

(ii)Testing corpus: On the other hand, 100 different sentences from the same location uttered by 10 different female speakers (10*100=1000 sentences). These are used as testing corpus. Testing and training corpus sentences are different.

5.3: Feature extraction and phonemes

At first we have to take sound wave and extract pertinent data from it by performing speech recognition. The sound wave is divided in time slices from which spectral features are extracted and the signal processing is start. At the beginning sound wave is in .wav format. It is converted to MFCC later.

5.4: Steps of the Experiments (Data preparation)

1. Create a folder named BanglaSpeech (any name can be given) into Drive D (any drive can be used).
2. Into the BanglaSpeech folder put the coder.scp, config.txt and HCopy.exe files.
3. We have to write the information’s in coder.scp (here is 3000 input speech which contains in 30 different folders and every folder got 100 inputs voice). After running HCopy command the input is on the left side and output is on the right side. And it looks like this:

“D:\\ BanglaSpeech\\Recorded_Audio_0.wav” “D:\\BanglaSpeech\\Recorded_Audio_0.mfc”
“D:\\ BanglaSpeech\\Recorded_Audio_1.wav” “D:\\BanglaSpeech\\Recorded_Audio_1.mfc”
“D:\\ BanglaSpeech\\Recorded_Audio_2.wav” “D:\\BanglaSpeech\\Recorded_Audio_2.mfc”
“D:\\ BanglaSpeech\\Recorded_Audio_3.wav” “D:\\BanglaSpeech\\Recorded_Audio_3.mfc”
“D:\\ BanglaSpeech\\Recorded_Audio_4.wav” “D:\\BanglaSpeech\\Recorded_Audio_4.mfc”
“D:\\ BanglaSpeech\\Recorded_Audio_5.wav” “D:\\BanglaSpeech\\Recorded_Audio_5.mfc”

4. Now HList.exe is stored into the BanglaSpeech folder. Then run our second command of HList. We will get something like this:

```
---------------------------- Source: Recorded_Audio_000.mfc ----------------------------
  Sample Bytes:  78        Sample Kind:   MFCC_D_A_C_K_0
  Num Comps:     39        Sample Period: 10000.0 us
  Num Samples:   208       File Format:   HTK
------------------------------------ Samples: 0->-1 ------------------------------------
0:      -8.039  -1.149  -0.432   4.899  -6.631  -2.435   0.932  -6.110   2.047  -1.275
         0.651   3.677  26.986  -0.319  -0.906   0.230  -1.591   1.961   0.101  -0.298
         3.750   0.629   0.167   0.199  -1.155   5.913   0.016  -0.050   0.373  -0.211
        -0.019   0.086   0.100   0.042  -0.675  -0.282  -0.228   0.169  -0.168
1:      -8.019  -3.430  -1.779   2.572  -3.292  -1.444  -0.127   3.678  11.110   3.228
        -0.394  -0.744  45.129  -0.510  -1.452   0.696  -2.219   2.493   0.663   0.509
         4.172  -1.419  -0.924  -1.432  -1.210   6.385   0.056  -0.027   0.370  -0.179
        -0.496  -0.172  -0.393  -0.679  -1.147  -0.349   0.065   0.410  -1.097
2:      -9.643  -4.537   1.393  -1.894   1.504  -2.427  -0.027   7.748   0.661  -2.690
         2.168   0.113  47.480  -0.142  -0.881   1.865  -2.331   1.598   0.249  -0.203
         3.748  -1.722  -0.697  -0.127  -0.282   4.837   0.113   0.017   0.013   0.085
        -0.863  -0.412  -0.729  -1.219  -0.654   0.278   0.718   0.515  -1.642
3:      -9.784  -6.714   2.136  -2.800   1.766   0.875   3.959   7.823  -4.354  -5.188
        -7.266  -0.589  48.663  -0.127  -1.052   1.264  -2.118  -0.338  -0.832  -2.310
         0.356  -3.928  -1.147   0.687   0.456   0.964   0.114   0.165  -0.316   0.384
        -0.729  -0.327  -0.532  -1.322   0.269   0.281   1.012   0.369  -1.702
4:      -7.866  -3.911   6.935  -4.067  -1.172  -2.350  -2.128  10.557   1.167  -0.552
         3.454   2.189  49.406   0.056  -1.022   0.014  -1.217  -0.940  -1.213  -2.532
        -0.435  -1.385   1.669   2.727   0.586   0.416   0.014   0.152  -0.554   0.476
        -0.350   0.099   0.510  -0.874   0.554  -0.331  -0.048  -0.251  -1.042
5:      -9.545  -9.003   1.770  -6.933  -3.644  -5.642 -10.627   4.053  -8.783  -3.578
         2.399   0.500  48.986  -0.037  -0.557   0.040  -0.854   0.115  -0.243  -0.984
        -0.345  -0.241  -0.702   2.203   0.200   0.084   0.045   0.373  -0.160   0.531
         0.079   0.580   1.204  -0.096   1.092  -0.285  -0.694  -0.511  -0.190
6:      -9.485  -8.504   1.644  -5.915  -0.492  -5.231  -5.397   7.457  -4.051   4.852
        10.973   2.498  49.396  -0.117  -0.369  -0.295  -0.584  -0.377   0.448   1.685
        -0.273  -0.795  -2.574  -1.126  -1.409   0.067  -0.044   0.215   0.162   0.517
        -0.019   0.436   0.890  -0.296   0.265  -0.562  -0.966  -0.406  -0.071
7:      -9.162  -7.201   4.979  -6.145   2.003   1.100   0.672   7.645  -2.951 -11.402
        -0.012   0.255  49.089   0.185   0.486   0.617   0.219  -0.224   1.237   1.600
        -0.207   1.239  -0.450  -0.855  -1.103   0.187  -0.139  -0.113   0.006   0.545
        -0.110  -0.059   0.055  -0.173  -0.142   0.744  -0.033   0.251  -0.038
```

5. After getting these we have to cut the header text and left sided numbers. Then we will get the following output:

```
-8.039  -1.149  -0.432   4.899  -6.631  -2.435   0.932  -6.110   2.047  -1.275
 0.651   3.677  26.986  -0.319  -0.906   0.230  -1.591   1.961   0.101  -0.298
 3.750   0.629   0.167   0.199  -1.155   5.913   0.016  -0.050   0.373  -0.211
-0.019   0.086   0.100   0.042  -0.675  -0.282  -0.228   0.169  -0.168
-8.019  -3.430  -1.779   2.572  -3.292  -1.444  -0.127   3.678  11.110   3.228
-0.394  -0.744  45.129  -0.510  -1.452   0.696  -2.219   2.493   0.663   0.509
 4.172  -1.419  -0.924  -1.432  -1.210   6.385   0.056  -0.027   0.370  -0.179
-0.496  -0.172  -0.393  -0.679  -1.147  -0.349   0.065   0.410  -1.097
-9.643  -4.537   1.393  -1.894   1.504  -2.427  -0.027   7.748   0.661  -2.690
 2.168   0.113  47.480  -0.142  -0.881   1.865  -2.331   1.598   0.249  -0.203
 3.748  -1.722  -0.697  -0.127  -0.282   4.837   0.113   0.017   0.013   0.085
-0.863  -0.412  -0.729  -1.219  -0.654   0.278   0.718   0.515  -1.642
-9.784  -6.714   2.136  -2.800   1.766   0.875   3.959   7.823  -4.354  -5.188
-7.266  -0.589  48.663  -0.127  -1.052   1.264  -2.118  -0.338  -0.832  -2.310
 0.356  -3.928  -1.147   0.687   0.456   0.964   0.114   0.165  -0.316   0.384
-0.729  -0.327  -0.532  -1.322   0.269   0.281   1.012   0.369  -1.702
-7.866  -3.911   6.935  -4.067  -1.172  -2.350  -2.128  10.557   1.167  -0.552
 3.454   2.189  49.406   0.056  -1.022   0.014  -1.217  -0.940  -1.213  -2.532
-0.435  -1.385   1.669   2.727   0.586   0.416   0.014   0.152  -0.554   0.476
-0.350   0.099   0.510  -0.874   0.554  -0.331  -0.048  -0.251  -1.042
-9.545  -9.003   1.770  -6.933  -3.644  -5.642 -10.627   4.053  -8.783  -3.578
 2.399   0.500  48.986  -0.037  -0.557   0.040  -0.854   0.115  -0.243  -0.984
-0.345  -0.241  -0.702   2.203   0.200   0.084   0.045   0.373  -0.160   0.531
 0.079   0.580   1.204  -0.096   1.092  -0.285  -0.694  -0.511  -0.190
-9.485  -8.504   1.644  -5.915  -0.492  -5.231  -5.397   7.457  -4.051   4.852
10.973   2.498  49.396  -0.117  -0.369  -0.295  -0.584  -0.377   0.448   1.685
-0.273  -0.795  -2.574  -1.126  -1.409   0.067  -0.044   0.215   0.162   0.517
-0.019   0.436   0.890  -0.296   0.265  -0.562  -0.966  -0.406  -0.071
-9.162  -7.201   4.979  -6.145   2.003   1.100   0.672   7.645  -2.951 -11.402
-0.012   0.255  49.089   0.185   0.486   0.617   0.219  -0.224   1.237   1.600
-0.207   1.239  -0.450  -0.855  -1.103   0.187  -0.139  -0.113   0.006   0.545
-0.110  -0.059   0.055  -0.173  -0.142   0.744  -0.033   0.251  -0.038
```

6. We have created Model to Model 10 folders. Inside model we get macro, proto_5states_39dim, vfloors, inside Model1 there is hmmdefs, macro, proto_5states_39dim. And inside Model2 to Model10 there remains hmmdefs and macro.

7. While working with model 1 we need to copy 53 words from monophone0 file.

5.5: Steps of the Experiments (Training)

Now we have to create mix 1, mix 2, mix 4, mix 8, mix 16 and mix 32 directories. The process is given below:

5.5.1: Mixture 1

1. After creating Model2 directory we run the following command.

HERest -T 1 -C HTK 39dpf.config -S Train.scp -I Train.MLF -L Label -M Model1 -H macro -w 3 -v 0.05 -t 250.0 150.0 1000.0 -H Model l \ hmmdefs -H Modell \macro -M Model 2 monophones .

Here two files are created under Model2 directory:

- Macro
- Hmmdefs

2. After creating Model 3 directory we run the following command.

HERest -T 1 -C HTK 39dpf.config -S Train.scp -I Train.MLF -L Label -M Model2 -H macro -w 3 -v 0.05 -t 250.0 150.0 1000.0 -H Model 2 \ hmmdefs -H Model2 \macro -M Model3 monophones.

Here two files are created under Model2 directory:

- Macro
- Hmmdefs

3. Until Model 10 directory is created we have to repeat the above step.

5.5.2: Mixture 2

1. After creating Mix2 directory and Model1 directory under Mix2 directory we run the following command.

HHEd -H Model10\macro -H Model10\hmmdefs -M Mix2\Model1 Hed\mix2.hed monophones .

Here two files are created under Model 1 directory:

- Macro
- Hmmdefs

2. Until Model 10 directory is created we have to repeat the above step.

5.5.3: Mixture 4

1. After creating Mix4 directory and Model1 directory under Mix4 directory we run the following command

HHEd -H Mix2 nModel10\macro -H Mix2nModel10\hmmdefs -M Mix4\Model1 Hed\mix4.hed monophones.

Here two files are created under Model1 directory.

- Macro
- Hmmdefs

2. Until Model 10 directory is created we have to repeat the above step.

5.5.4: Mixture 8

1. After creating Mix8 directory and Model1 directory under Mix8 directory we run the following command.

HHEd -H Mix4 nModel10nmacro -H Mix4nModel10\hmmdefs -M Mix8\Model1 Hed\mix8.hed monophones.

Here two files are created under Model1 directory.

- Macro
- Hmmdefs

2. Until Model10 directory is created we have to repeat the above step.

5.5.5: Mixture 16

1. After creating Mix16 directory and Model1 directory under Mix16 directory we run the following command.

HHEd -H Mix8 nModel10nmacro -H Mix8\Model10\hmmdefs -M Mix16 \ Model1Hed\mix16.hed monophones:

Here two files are created under Model1 directory.

- Macro
- Hmmdefs

2. Until Model10 directory is created we have to repeat the above step.

5.5.6: Mixture 32

1. After creating Mix32 directory and Model1 directory under Mix32 directory we run the following command.

HHEd -H Mix16 \Model10nmacro -H Mix16\Model10\hmmdefs -M Mix32nModel1 Hed\mix32.hed monophones.

Here two files are created under Model1 directory

- Macro
- Hmmdefs

2. Until Model10 directory is created we have to repeat the above step.

5.5.7:Text Data to HTK Data Conversion

We used visual studio dot net 2005 for converting the following code.

```
#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 39
#define BIN 39
#define MAX_FNAME 512

#define IN_SCRIPT "FileList.txt"
#define OUT_SCRIPT "test.scp"

typedefstruct 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, intfnum, intch);

void main(void)
{
	int n;
	HISTOGRAM*hist;
	charfname[MAX_FNAME]={'\0'};
	char Path[MAX_FNAME]={'\0'};
	//char LPath[MAX_FNAME]={'\0'};
	char temp[6]={'\0'};
	chartmp[10]={'\0'};
	//char phone[6]={'\0'};
	intfnum=0;
	FILE *fp,*fp1,*fp3;
	double data;
	intk,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\n");
		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);
}
}

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

```
            strcpy(Path,"");
            fscanf(fp1,"%s",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;
}

voidWriteWave(double **vec, char *fname, intfnum, intch)
{
        FILE *fp;
        int f, c, n;

        intr_fnum;
```

```
intsp = 100000;                    /* 10ms: HTK,Å,Í100ns’PˆÊ,ÅƒJƒEƒ“ƒg */
shortintsampSize;
shortintsampKind = 9;                    /*                    ƒ†□ [ƒU:
HTK,Åƒ†□ [ƒU,ðŽ¦,·ƒR□ [ƒh  */
//short intsampKind = 0x0B46;
//unsigned char temp[1000000];
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];
floatdatabuf[FILTER_NUM];
float **floatbuf;

/* ˆêŽžƒoƒbƒtƒ@,ÌŠm•Û, 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(OUTPUT_DIM, sizeof(float))) == NULL)
	{
	if ((floatbuf[n] = (float *)calloc(FILTER_NUM, sizeof(float))) == NULL)
	{
		exit(1);
	}
}

/* ˆêŽžƒoƒbƒtƒ@,ÌŠm•Û, 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);
```

```
/* ƒoƒCƒgƒXƒ□ ƒbƒv‚Ì‚½‚ß‚Ìˆêžž□ o—Í */
/* output for temporal byte swap  */
if ((fp= fopen(fname, "wb")) == NULL)
{
fprintf(stderr, "\nfile 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);

/* ƒoƒCƒgƒXƒ□ ƒbƒv‚Ì‚½‚ß‚Ì□ Ä“ü—Í */
/* re-input for byte swap  */
if ((fp= fopen(fname, "rb")) == NULL)
{
fprintf(stderr, "\nfile 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);

/* ƒtƒŒ • [ƒ€ƒTƒCƒYƒXƒ□ ƒbƒv, 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);

/* ƒTƒ“ƒvƒŠƒ“ƒOŽüŠú, 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);

/* ƒxƒNƒgƒ‹ƒTƒCƒY, 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);

/* ”gŒ`‚ÌŽí—Þ, 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);


/* “Á’¥ƒxƒNƒgƒ‹ƒXƒ□ ƒbƒv, swapof 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;
        }
}

/* ƒoƒCƒgƒXƒ□ ƒbƒv□ Ï,Ý,Ìƒf□ [ƒ^,ð□o—Í */
/* output into file after byte swap */
if ((fp= fopen(fname, "wb")) == NULL)
{
        fprintf(stderr, "\nfile 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);

/* ˆêŽžƒoƒbƒtƒ@,ÌŠJ•ú, deallocation of temporal buffer */
for (n = 0; n < MAX_FRAME_NUM; n++)
{
        free(floatbuf[n]);
}
```

```
        free(floatbuf);
        free(temp);
}
```

5.6: Steps of the Experiments (Testing)

1. Take all testing files.
2. Replace the genoutput path to the filelist.txt file.
3. In the filelist.txt replace the .wav by .gen.
4. Copy and paste the filelist.txt to the convert.c program.
5. Rename filelist.txt to source.acp.
6. Create the test folder and run convert.c program to the Visual Studio 6.
7. Create a convertT folder at Visual studio 6.
8. Copy convert folder and filelist.txt to the convertT folder.
9. Copy BanglaFemaleTestVoice folder to the convertT folder.
10. Create a Test named folder.
11. Now run the convert.c code and test.scp create.
12. Get the path of test folder and paste at test.scp
13. Bring the test.scp file to the BanglaSpeechTrain folder.
14. Paste test.scp,test folder, Dictionary.dic, grammer.grm, HTK_22dpf.config, Lattice.lat,test.mlf, monophones0, proto_5states_22dim.txt to the MFCC Test folder.
15. Now run the HVite command.
16. Command is:(HVite -T 1 -C HTK_22dpf.config -S Train.scp -t 300.0 -s 5.0 -i ./Results/Mix1model10.txt -w lattice.lat -H model10/hmmdefsDictionary.dic monophones0).
17. Next create the Models(model1 – model10)
18. And create MixFolders(mix1 – mix16) and create model10 into each MixFolder.
19. Then we run HERest command.
20. Command : HERest -T 1 -C HTK_22dpf.config -S Train.scp -I Train.mlf -L Train -w 3 -v 0.05 -t 250.0 150.0 1000.0 -H model9/hmmdefs -H model1/macro -M model10 monophones0.
21. Next Run HVite command (HVite -T 1 -C HTK_22dpf.config -S Train.scp -t 300.0 -s 5.0 -iTrainResult/mix4model10.txt -w Lattice.lat -H Mix4/Model1/hmmdefsDictionary.dic monophones0)
22. Next Run HParse(HParse.exegrammer.grmLattice.lat).
23. And then Run HCompV (HCompV -T 1 -C HTK_22dpf.config -S Train.scp -m -v 0.01 -f 0.01 -M ./Model proto_5states_22dim.txt).
24. Next Run HRest (HRest -T 1 -C HTK_22dpf.config -S Train.scp -I Train.mlf -L Train -M Model1 -H model/macro -w 3 -v 0.05 -i 40 -m 1 model/proto_5states_22dim.txt).
25. Next Run HVite (HVite -T 1 -C HTK_22dpf.config -S Test.scp -t 300.0 -s 5.0 -i Result/mix1model10.txt).

26. Next Run HResult(HResult -I test.mlf -L Test monophones0 Result/mix1model10.txt>Result/mix1result.txt).
27. Again run HVite (HVite -T 1 -C HTK_22dpf.config -S Train.scp -t 300.0 -s 5.0 -iTrainResult/mix2model10.txt -w Lattice.lat -H Mix2/Model10/hmmdefsDictionary.dicmonophones0) and rename mix1 as mix2.
28. Run HResult (HResults -I test.mlf -L Test monophones0 Result/mix2model10.txt>Result/mix1result.txt) and rename mix1 as mix2.
29. This process continues till mix16.
30. And we get the result to the Result folder into "mix1result.txt" and more.

5.7: Results of the Experiment

Table 9: Experiment Result – 1/4

Close Test (3000 training, 3000 testing)

	Mix 1	Mix 2	Mix 4	M ix 8	Mix 16	Mix 32
Phoneme correct rate	40.97%	51.33%	54.05%	55.84%	57.53%	57.68%
Phoneme accuracy	22.55%	34.25%	37.26%	39.82%	42.48%	43.16%

Table 10: Experiment Result – 2/4

Close Test (3000 training, 3000 testing)

	Mix 1	Mix 2	Mix 4	Mix 8	Mix 16	Mix 32
Insertion, I	14896	13813	13579	12960	12175	11741
Deletion, D	10619	8878	8650	8873	8490	8679
Substitution, S	37127	30487	28517	26843	25861	25552
Total, N	80880	80880	80880	80880	80880	80880
Correct, H	33134	41515	43713	45164	46529	46649

Table 11: Experiment Result – 3/4

Open Test (3000 training, 1000 testing)

	Mix 1	Mix 2	Mix 4	Mix 8	Mix 16
Phoneme correct rate	35.57%	41.70%	44.03%	46.79%	48.87%
Phoneme accuracy	11.65%	18.91%	22.45%	25.29%	27.27%

Table 12: Experiment Result – 4/4

Open Test (3000 training, 3000 testing)

	Mix 1	Mix 2	Mix 4	Mix 8	Mix 16
Insertion, I	5455	5198	4922	4906	4926
Deletion, D	2223	2287	2328	2218	2118
Substitution, S	12475	11012	10439	9919	9546
Total, N	22811	22811	22811	22811	22811
Correct, H	8113	9512	10044	10674	11147

In our experiment we can see that the accuracy is increasing at first, but when the mixture level is getting higher the accuracy level is increased so slowly. In open text and close text we see almost the same problem. That is why we took the result of mix 16 in the open test and mix 32 in the close test as a good mixture.

Chapter 6

Conclusion and Future Work

6.1. Conclusion

In the overall thesis work we developed a Bangla Automatic Phoneme Recognition for our Bangla Language. We are trying to improve the performance of Bangla voice interface of the human computer. This paper using a huge data of Bangla speech corpus and do some experiments and using many new Tools.

We can learn the following conclusion from this experiment:

(i) **Accuracy Increases with the mixer components:** When the mixer is increased the accuracy is also be increased.

(ii) **Mixer 32 given better result within mixer investigated:** Above the experiment when the number of mix is increased the accuracy is also be increased. Mixer 32 given the better result. When the mixer is increased the accuracy is increased but any certain time it will be neutral.

(iii) In this project, the main focus was to develop an automatic phoneme recognizer for Bangla language. We have learned a great thing in this project.

1) The MFCC-based system provides extraordinary improvement of Bangla phoneme recognition accuracy for both training and test data.

2) A higher Bangla phoneme correct rate for training and test data is also obtained by the MFCC-based system.

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

4) If we have large vocabularies, we will not have enough training samples for the creation of word models.

6.2. Future work

Some topic is out of this thesis. In future we like to do this work such as:

(i) Inclusion of context information: We want to increase the number of context. If we tested more and more data than the result will be accurate.

(ii) Phoneme probability estimation by Machine Learning: We want to use machine learning for estimation phoneme probability.

References:

1. Z. Mengjie, “Overview of speech recognition and related machine learning

Techniques,” Technical Report, December 2004.

2. A. Dix, J. Finlay, G. Abowd, and R. Beale, Human-Computer Interaction. Prentice

Hal, Englewood Cliffs and NJ and USA, 1998.

3. R. John, W. Y. Wong, C. Chung, and D. K. Kim, “Automatic speech recognition

For generalized time-based media retrieval and indexing,” sixth ACM International

Conference on Multimedia , 1998.

4. Jelinek, “The development of an experimental discrete dictation recognizer,” IEEE, 73,

1985.

5. R. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, and V. Zue, Survey of the State of the Art in Human Language Technology. Cambridge University Press, 1996.