CSE-4000: Project/Thesis

Bangla Speech Corpus For Large Vocabulary ASR System

By:

Sujoy Datta
Roll: 0907011
&
Anurag Bhattacharjee
Roll: 0907061

Supervisor:

Dr. Pintu Chandra Shill
Assistant Professor
Dept. of Computer Science and Engineering
Khulna University of Engineering and Technology

Department of Computer Science and Engineering
Khulna University of Engineering and Technology
Khulna- 9203, Bangladesh.

Abstract:

Speech recognition and understanding of spontaneous speech have been a goal of research since 1970. But research and development on Large Vocabulary Bangla ASR System does not see that much light because there is lack of Bangla speech or language corpus. The property of Bangla language is that it is very rich. This thesis paper represents the Bangladeshi Bangla speech database, a modern standard Bangla speech corpus composed of utterances pronounced by 15 speakers selected from different regions of Bangladesh. Speech Corpora is the collection of speech and text. It is the backbone of ASR system. We have implemented bigram modeling for phonemes for the betterness of language modeling. This thesis paper also reports the results of the Automatic Speech Recognition (ASR) application of the corpus and outlines an original global monophone recognition model designed to handle linguistic variability. The word recognition rate for this ASR reference system is so so and may constitute a useful baseline ASR system for our mother tongue- Bangla.

Acknowledgment

All praises and greatness goes to God for His limitless kindness & blessings. Without His desire we would not be here as we are today.

We are grateful to our honorable supervisor **Dr. Pintu Chandra Shill** sir (Assistant Professor, Department of Computer Science & Engineering, KUET).He gave us moral support and guided us in different matters regarding the topic. He had been very kind and patient while suggesting us the outlines of this thesis and correcting our doubts. We thank him for his overall supports.

We express our deep sense of appreciation and sincere gratitude to **Dr. Md. Faijul Amin** sir, for introducing us this promising research area. We are heartily thankful to him for his unlimited encouragement, full support and invaluable advices from the initial, which enabled us to develop an understanding of the subject.

Besides we would like to acknowledge to our respected seniors, friends who shared their views and give their kind association while accomplishing the thesis.

Last but not the least, we are also thankful to my family, friends and mates who have rendered their whole hearted support at all times for the successful completion of this Thesis.

Any constructive comments, suggestions, criticism from teachers as well as seniors will be highly appreciated and gratefully acknowledged.

Contents

List of Figures

Chapter 1
INTRODUCTION

Speech is the most natural means of communication between humans; it can be done without any tools or any explicit education. It is one of the first skills we learn to use. Babies quickly learn how to react to the voice of their mother and they even quicker learn to produce noise when they are in need. Speech is also the most important way of communicating. It has always been; before mankind invented writing, the spoken word was the only way of passing knowledge. Despite all our novel ways of communicating, like e-mail and chat, speech is still the number one means of communication, a fact once again proven by the immense popularity of cellular phones.

Speech recognition is a topic that is very useful and important in our daily environment. Generally Speech recognizer is a machine that understands human and their spoken word in some way and acts thereafter. It is also known as "Automatic Speech recognition" ASR. These systems analyze the person's specific voice and use it to fine tune the recognition of that person's speech, resulting in more accurate transcription. It can be used, for example, in a car environment to voice control over non critical operations such as dialing a phone[1].

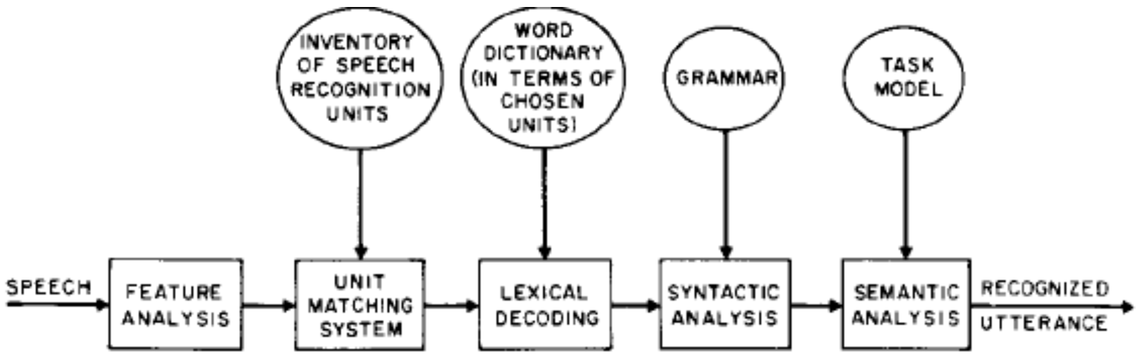


Figure 1.1: Automatic Speech Recognition

Creating of a speech corpus is a mandatory task for building and testing any speech recognition system. The performance of a recognizer widely depends on the speech corpus. There are various reasons that can change the performance of ASR system such as utterance, dialect etc. It is very much important to benchmark the system with standard database, which covers a good range of variability[3].

1.1: Previous work:

In the last 30 years researchers from areas like psychology, linguistics, electrical engineering and computer science have worked on English speech recognition. While the first systems could only differentiate between 'yes' and 'no', currently, speaker independent systems exist with a vocabulary of over 60000 words that can recognize continuous speech (that is complete sentences or paragraphs) with an accuracy of 95% or higher. Morever, speech recognition system for different language is developing for years[4].

Unfortunately, only a very few works have been done for ASR in Bangla (can also be termed as Bengali), which is one of the largely spoken languages in the world. More than 220 million people speak in Bangla as their native language. It is ranked sixth based on the number of speakers[17].

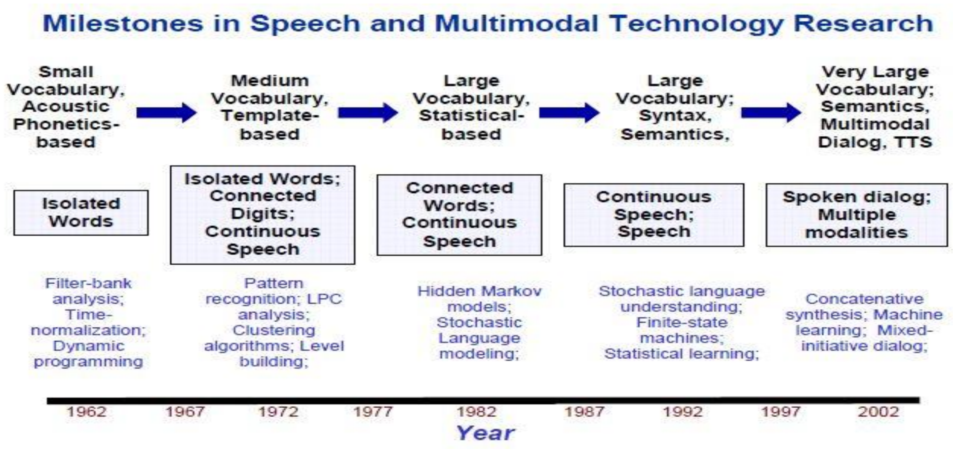

Milestones in Speech and Multimodal Technology Research
Small Vocabulary, Acoustic Phonetics-based
Medium Vocabulary, Template-based
Large Vocabulary, Statistical-based
Large Vocabulary; Syntax, Semantics,
Very Large Vocabulary; Semantics, Multimodal Dialog, TTS
Isolated Words
Isolated Words; Connected Digits; Continuous Speech
Connected Words; Continuous Speech
Continuous Speech; Speech
Spoken dialog; Multiple modalities
Filter-bank analysis; Time-normalization; Dynamic programming
Pattern recognition; LPC analysis; Clustering algorithms; Level building;
Hidden Markov models; Stochastic Language modeling;
Stochastic language understanding; Finite-state machines; Statistical learning;
Concatenative synthesis; Machine learning; Mixed-initiative dialog;
1962
1967
1972
1977
1982
1987
1992
1997
2002
Year

Figure 1.2: History of Speech Recognition

For having better performance of the speech recognizers, it is inevitable to have speech corpus of that particular language. The first initiative was taken by MIT to create TIMIT speech corpus. Speech corpus has been developed in many other foreign countries. Although some Bangla speech databases for the eastern area of India (West Bengal and Kolkata as its capital) were developed, but most of the natives of Bangla (more than two thirds) reside in Bangladesh, where it is the official language. Besides, the written characters of Standard Bangla in both the countries are same; there are some sounds that are produced variably in different pronunciations of Standard Bangla, in addition to the myriad of phonological variations in non-standard dialects [6]. Therefore, there is a need to do research on the main stream of Bangla, which is spoken in Bangladesh, ASR[2].

Some developments on Bangla speech processing or Bangla ASR can be found. For example, Bangla vowel characterization is done; isolated and continuous Bangla speech recognition on a small dataset using HMMs is described. But there is no large vocabulary Bangla ASR system corpus.

1.2: Our Work:

There can be a number of reasons that can change the performance of a speech recognition system. The reasons are like session variability, intra-speaker and inter-speaker variability and instrument variability. A speaker’s utterance can be affected by a number of variations like regional, social, stylistic and individual ones. It is important to benchmark the system with standard database, which covers a good range of variability. We have used many books written by Dr. Md. Jafor Iqbal, Bivutivushon Bandapaddhay etc. Text corpora consist of 600 unique sentences, 23000 unique words. Each sentence was recorded with 16000Hz sample frequency, 16-bit sample rate mono sound. Speech signals are recorded with A4Tech headphone with recording device. Dictionary is a phonetic representation of words. Dictionary was created with object-oriented programming. After this, dictionary was corrected manually.

We have collected data from various regional people. in room environment. The contributed people are from Dhaka, Chittagong, Rajshahi, Sylhet, Barishal, Mymensingh, and Khulna. They have different dialect and utterance for the same sentence. Voice source features like fundamental frequency, format frequency, Jitter Shimmer, voice onset time and harmonic-to-noise ratio changes significantly from place to place. We also implement bigram models for phoneme in language modeling. Our built corpus can be used for phoneme-base and word-based speech recognition model.

Checking a corpus is an important task in speech recognition. It evaluates the performance and effectiveness of a speech corpus. There are many tools to do such work. CMUSPHINX is one of them. We learn about this toolkit. We installed this in Linux operating system and learned its functionality. After building the Bangla speech corpus, we built up the trainer using SphinxTrainer. Then this training module is passed to Sphinxbase for training to the system. The system has been tested with PocketSphinx.

1.3: Method

Speech corpus has different parts. It consists of speech samples, dictionary, phone list, word list and transcription file. Speech samples are simple .WAV file. Dictionary presents phonetic transcription of sorted words. We have produced the pronunciation dictionary manually because same word can be pronounced differently. Word list consists of all the words in increasing word. We have included 3 files for the easiness of language modeling. We have included a phoneme distribution file that keeps the record of no. of occurrence of a phoneme in the. We design a bigram occurrence file that keeps the record of no. of occurrence of simultaneous two phonemes. We included a bigram model file that is used to store bigram probability of phonemes.

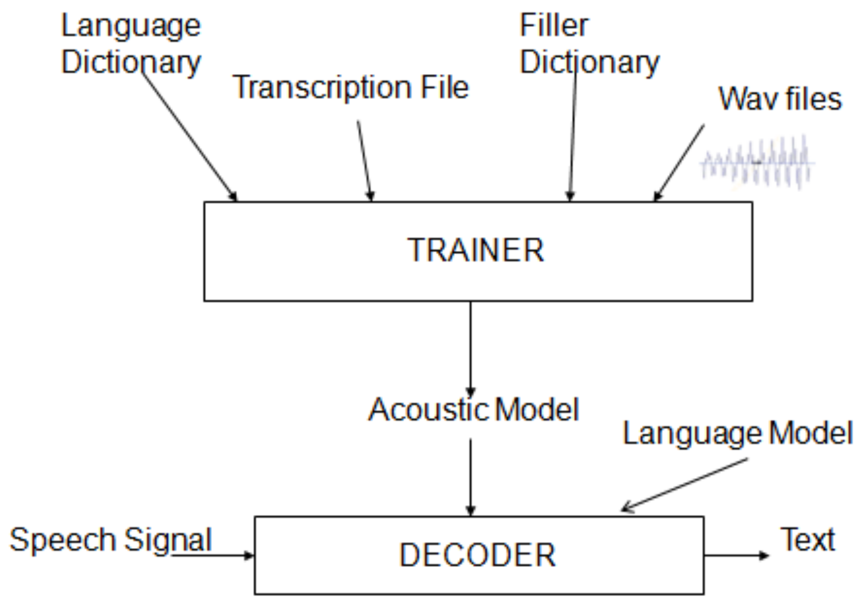


Figure 1. 3:The Model

1.4: Objectives of this Thesis

The main objectives of this thesis are:

- To study on speech recognition system and gather knowledge on how the system and its various components works.
- To build a large vocabulary speech corpus for Bangla language using variation and many people and try to eliminate speech corpus impurity.
- To check the performance of the speech corpus using appropriate speech recognition toolkit.

1.5. Organization of the Thesis

This thesis report consists of-

- Chapter 2 discusses the structure of a Large Vocabulary Automatic Speech Recognition and different speech recognition toolkit.
- Chapter 3 presents the Bangla Speech Corpus building procedure and the whole process of installing training and testing in Sphinx.
- Chapter 4 discusses the results and cause of deviation of the result.
- Conclusions and scope of future works are given in Chapter 5.

Chapter 2: Background

This chapter will describe the pros and cones of speech recognition system, speech corpus as a whole and about the toolkit we used for recognition.

2.1 Approaches of Speech Recognition

There are many approaches or algorithms for speech recognition. The approaches are given below

2.1.1 Hidden Markov Model (HMM):

The Hidden Markov model (HMM) is a very powerful mathematical tool for modeling time series. It provides efficient algorithms for state and parameter estimation, and it automatically performs dynamic time warping for signals that are locally squashed and stretched. It can be used for many purposes other than acoustic modeling. Hidden Markov models are based on the well-known Markov chains from probability theory that can be used to model a sequence of events in time. Hidden Markov models are based on the well-known Markov chains from probability theory that can be used to model a sequence of events in time. The topology of the network shows an important property of Markov chains, namely that the next state only depends on the current state the model is in, regardless of how it got in the current state; this property is often referred to as the Markov property[5].

In the Markov model each state corresponds to one observable event. But this model is too restrictive, for a large number of observations the size of the model explodes, and the case where the range of observations is continuous is not covered at all. Therefore the Hidden Markov concept extends the model by decoupling the observation sequence and the state sequence. For each state a probability distribution is defined that specifies how likely every observation symbol is to be generated in that particular state. As each state can now in principle generate each observation symbol it is no longer possible to see which state sequence generated a observation sequence as was the case for Markov models, the states are now hidden, hence the name of the model.

2.1.2 Dynamic Time Wrapping (DTW)

Dynamic time warping is an approach that was historically used for speech recognition but has now largely been displaced by the more successful HMM-based approach. Dynamic time warping is an algorithm for measuring similarity between two sequences that may vary in time or speed. For instance, similarities in walking patterns would be detected, even if in one video the person was walking slowly and if in another he or she were walking more quickly, or even if there were accelerations and decelerations during the course of one observation. DTW has been applied to video, audio, and graphics – indeed, any data that can be turned into a linear representation can be analyzed with DTW[6].

A well-known application has been automatic speech recognition, to cope with different speaking speeds. In general, it is a method that allows a computer to find an optimal match between two given sequences (e.g., time series) with certain restrictions. That is, the sequences are "warped" non-linearly to match each other. This sequence alignment method is often used in the context of hidden Markov models.

2.1.3 Neural Network

Neural networks emerged as an attractive acoustic modeling approach in ASR in the late 1980s. Since then, neural networks have been used in many aspects of speech recognition such as phoneme classification, isolated word recognition, and speaker adaptation[7].

In contrast to HMMs, neural networks make no assumptions about feature statistical properties and have several qualities making them attractive recognition models for speech recognition. When used to estimate the probabilities of a speech feature segment, neural networks allow discriminative training in a natural and efficient manner. Few assumptions on the statistics of input features are made with neural networks. However, in spite of their effectiveness in classifying short-time units such as individual phones and isolated words, neural networks are rarely successful for continuous recognition tasks, largely because of their lack of ability to model temporal dependencies. Thus, one alternative approach is to use neural networks as a pre-processing e.g. feature transformation, dimensionality reduction, for the HMM based recognition.

2.2 Why HMM is used in Speech recognition:

Modern general-purpose speech recognition systems are based on Hidden Markov Models. These are statistical models that output a sequence of symbols or quantities. HMMs are used in speech recognition because a speech signal can be viewed as a piecewise stationary signal or a short-time stationary signal. In a short time-scale (e.g., 10 milliseconds), speech can be approximated as a stationary process. Speech can be thought of as a Markov model for many stochastic purposes[4].

Another reason why HMMs are popular is because they can be trained automatically and are simple and computationally feasible to use. In speech recognition, the hidden Markov model would output a sequence of *n*-dimensional real-valued vectors (with *n* being a small integer, such as 10), outputting one of these every 10 milliseconds. The vectors would consist of kestrel coefficients, which are obtained by taking a Fourier transform of a short time window of speech and decorrelating the spectrum using a cosine transform, then taking the first (most significant) coefficients. The hidden Markov model will tend to have in each state a statistical distribution that is a mixture of diagonal covariance Gaussians, which will give likelihood for each observed vector. Each word, or (for more general speech recognition systems), each phoneme, will have a different output distribution; a hidden Markov model for a sequence of words or phonemes is made by concatenating the individual trained hidden Markov models for the separate words and phonemes.

2.3 Speech recognition basics

Speech recognition (SR) in terms of machinery is the process of converting an acoustic signal, captured by a microphone or a telephone, to a set of words. It is a broad term which means it can recognize almost anybody's speech, but to make the machine independent of voice, huge training data is required[11]. There are basically two types of SR:

1. Isolated speech recognition - ISR

2. Automatic speech recognition - ASR

An isolated-word speech recognition system requires that the speaker pause briefly between words, whereas a continuous speech recognition system does not. The Automatic speech consists of continuous utterance which is representative of real speech. On the other hand a sentence constructed from connected words does not represent real speech as it is actually concatenation of isolated words. For Isolated word the assumption is that the speech to be recognized comprises a single word or phrase and to be recognized as complete entity with no explicit knowledge or regard for the phonetic content of the word or phrase.

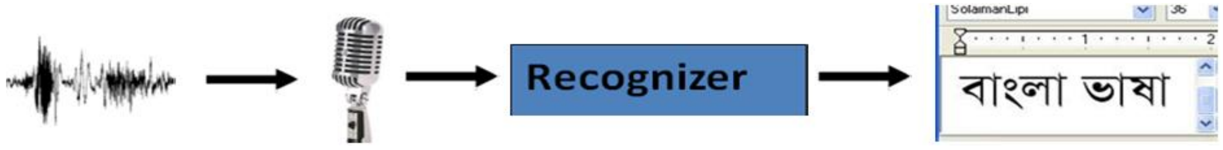


Figure 2.1: Simple Overview of Speech recognition System

To understand speech recognition technology, some of the terms that are used throughout the full report and needed to know are:

1. Utterance: An utterance is the vocalization (speaking) of a word or words that represent a single meaning to the computer. Utterances can be a single word, a few words, a sentence, or even multiple sentences.

2. Speaker Dependence: Speaker dependent systems are designed around a specific speaker. They generally are more accurate for the correct speaker, but much less accurate for other speakers. They assume the speaker will speak in a consistent voice and tempo. Speaker independent systems are designed for a variety of speakers. Adaptive systems usually start as speaker independent systems and utilize training techniques to adapt to the speaker to increase their recognition accuracy.

3. Vocabularies: Vocabularies (or dictionaries) are lists of words or utterances that can be recognized by the SR system. Generally, smaller vocabularies are easier for a computer to recognize, while larger vocabularies are more difficult. Unlike normal dictionaries, each entry doesn't have to be a single word. They can be as long as a sentence or two. Smaller vocabularies can have as few as 1 or 2 recognized utterances (e.g."Wake Up"), while very large vocabularies can have a hundred thousand or more.

4. Training: Process of learning the characteristics of sound units is called Training. The trainer learns the parameters of the models of sound units using a set of sample speech signals called training database (TD)

5. A Language Dictionary: Accepted Words in the Language are mapped to sequences of sound units representing pronunciation, sometimes includes syllabification and stress.

6. A Filler Dictionary: Non-Speech sounds are mapped to corresponding non-speech or speech like sound units

7. Phone: Way of representing the pronunciation of words in terms of sound units. The standard system for representing phones is the International Phonetic Alphabet or IPA. English Language use transcription system that uses ASCII letters where as Bangla uses Unicode letters

8. Language Model: assigns a probability to a sequence of m words by means of a probability distribution. To model it we can use a regular grammar.

2.4: Formulation and Components of Speech Recognition:

Mathematically we have to find the word sequence W^

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} P(\mathbf{W}|\mathbf{O})$$

Given the acoustic evidence O and word sequence W

But this formula is not directly computable. So another formulation is needed. From Bayes formula we get,

$$P(\mathbf{W}|\mathbf{O}) = \frac{P(\mathbf{W})\,P(\mathbf{O}|\mathbf{W})}{P(\mathbf{O})}$$

Ultimately we get,

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} P(\mathbf{W})\, P(\mathbf{O}|\mathbf{W})$$

So a speech recognizer consists of three components:

1. Acoustic processing: The first step in speech recognizer design is to decide what acoustic data **O** will be observed. Therefore a front end is needed that will transform the original waveform into a sequence of symbols $\boldsymbol{O_i}$ with which the recognizer will deal. Strictly speaking this is not necessary; speech recognition could be done by performing pattern recognition algorithms directly on the speech signal, as this signal contains all information. But as mentioned before there are many possible variations in a speech signal and visually similar waveforms do not necessarily indicate perceptually similar sounds. Therefore some preprocessing may be useful to reduce the amount of noise introduced by the environment and the recording hardware and to reduce correlation in the input signal and to extract relevant features.

2. Language Modeling: we are able to compute for every word sequence **W** the a priori probability *P*(**W**) that the speaker wishes to utter **W**. As **W** is a string, these probabilities can be decomposed as Bayesian network. The choice of $\mathbf{W_i}$ thus depends on the entire history of the discourse sofar. In reality it is impossible to estimate these probabilities even for moderate values of *i* as most of these histories would be unique. For a vocabulary of size |*V*| there are |*V*|i-1 different histories. For example even for a relatively small vocabulary of 1000 words and i=3 there would be one billion different histories.

3. Acoustic Modeling: The acoustic model determines what sounds will be produced when a given string of words is uttered. Thus for all possible combinations of word strings **W** and observation sequences **O** the probability *P*(**O**| **W**)must be available. This number of combinations is just too large to permit a lookup, in the case of continuous speech its even infinite. It follows that these probabilities must be computed on the fly, so a statistical acoustic model of the speakers' interaction with the recognizer is needed. The total process modeled involves the way the speaker pronounces the words of **W**, the ambience (room) noise, the microphone placement and characteristics and the acoustic processing performed by the signal processing front end.

2.5 Automatic Speech Recognition (ASR) Structure

The crossing order of the ASR system from the Speech data is evidenced in the figure 2.2[6]. The complete procedure is:

1) Speech databases are essential to allow to grow in performance. The speech data may come from databases or from microphone, since these data comply with different standards, the database interface processes data to supply the successive blocks with a consistent input.

2) The raw speech samples must be processed to extract significant data from recognition. In essence, sufficient statistics **Y** are extracted from the speech sample **X** by simply applying function **g: Y=g(X)**. Further processing is generally performed to adapt ASR to the environment variability and to reduce its influences.

3) The extracted features Y are suitable to initialize the hidden markov model.

4) HMM training of parameters related to phonemes are basically estimation of the probabilities. **P (y_il $s_{i,}$ϴ)** and **P(s_i l s_{i-1},ϴ)**.

5) HMM parameters relative to the words are estimated in various procedures

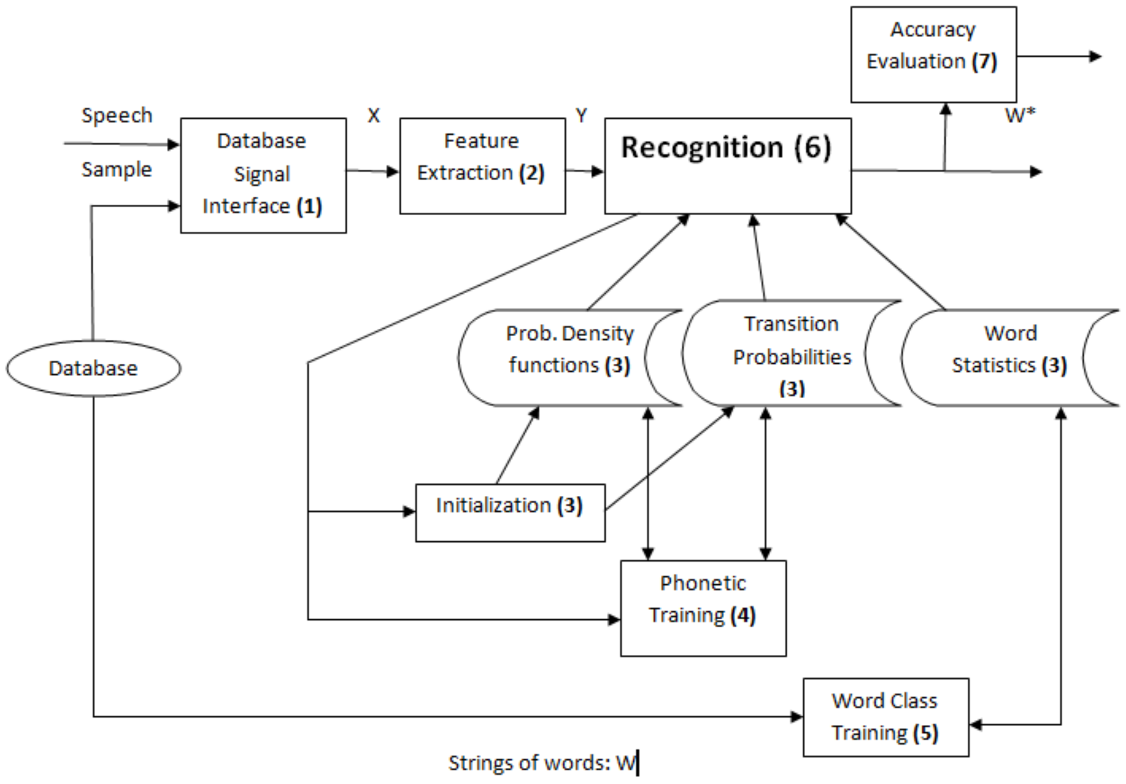


Figure 2.2: ASR Structure

6) After training the whole HMM, recognition can be performed to obtain the more likely sequence of words **W***. Viterbi or other search algorithms are considered to reduce search complexity.

7) Accuracy evaluation procedure is used to test the performance of the ASR system.

2.6 Speech Corpus

Speech corpora are the backbone of automatic speech recognition (ASR) system. For having better performance of the speech recognizers, it is inevitable to have speech corpus of that particular language. The first imitative was taken by MIT to create TIMIT speech corpus[21]. Speech corpus has been developed in many other foreign languages. Creation of a speech corpus is mandatory task for building and testing any speech recognition system. The quality of the recognizer depends on the quality of the speech corpus also. There can be a number of reasons that can drastically change the performance of a speech recognition system. The reasons are like session variability, inter-speaker variability and instrument variability. The speakers can utter same words in different manners during different session emotional state. This can affect the baseline performance of a speech recognition system very much[9]. Moreover another important issue is dialects of the language. It is very much important to bench-mark the system with a standard database which covers a good range of variability. A wide variety of languages belonging to different linguistics group are spoken in Bangladesh. In addition, different dialects are spoken in the same language depending on the region to which speaker does belong.

2.7 Speech Corpus Building

In the following figure-2.3, typical speech corpus production technique is given[8]. It is consists of –

2.7.1 Corpus Specification

This section gives an overview about the basic requirements of any speech corpus specification. There may be additional things to cover in the specs depending on the special nature of corpus. The high costs of speech corpus production can be optimally exploited by specifying as many diverse features into one speech collection as possible. For example in a telephone based corpus with the primary aim to recognize digits and numbers the overall costs will not dramatically increase with some additional non-prompted or even spontaneous recordings within the same recording sessions. However, the re-usability of the corpus will be much higher than with a corpus that only contains read digits and numbers. It also describes sampling rate, sampling frequency, number of channel format of all the data.

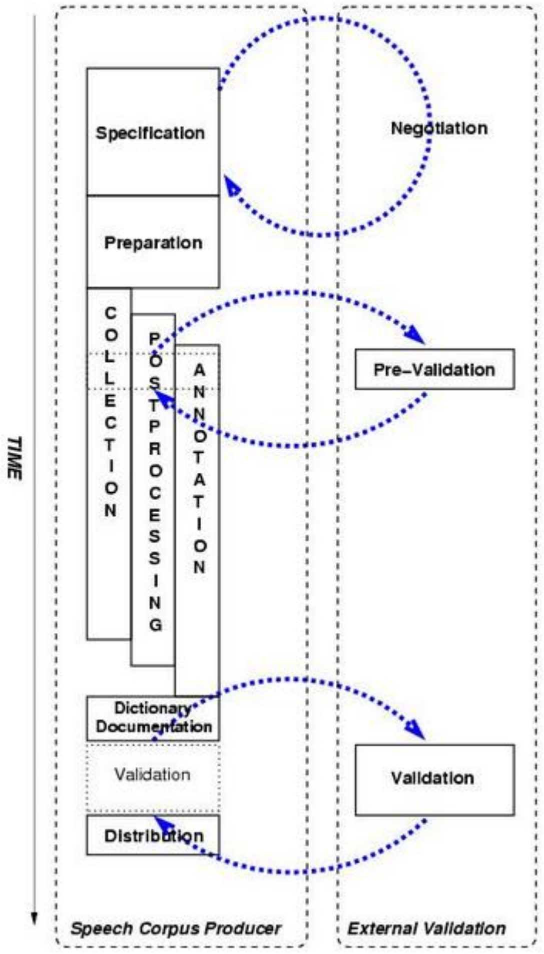


Figure 2.3: Typical Corpus Building Procedure

2.7.2 Preparation

After all have agreed on the specifications of the speech corpus one will need some time to prepare the collection phase. Do not underestimate the time required for this preparatory phase: very often it takes longer to prepare a data collection than to actually record the data.

2.7.3 Collection

Collection describes the activities at the core of every speech corpus production: the actual recording of the speech signals. Most of the technical and other advice one will found in the previous phase. One should have knowledge on Ongoing Documentation, Pre-Validation, Quality Control, Data Pipelining and Recruitment.

2.7.4 Post-processing

Post-processing includes all processing steps from the recorded raw signal data to the final distributed corpus. The following processing steps might not all be necessary in your corpus collection; however, some of them are: file transfer from recording device to computer, file name assignment, filtering, cutting, synchronization, re-sampling, format conversion, special conversion for annotation and automatic error detection. Please note that some of these processing steps may be applied after or between the annotations steps described in the next chapter depending on the structure of your data pipelining.

2.7.5 Annotation

The annotation of a speech corpus denotes all symbolic information that is directly related to the speech signal, either via the physical time scale, in which case we speak of a *segmentation and labeling*, or via some semantic content of the speech signal, in which case we speak of a *transcription or tagging*.

2.7.6 Pronunciation Dictionary

The pronunciation dictionary lists the most likely pronunciation or citation form of all words that are contained in the speech corpus. A pronunciation dictionary cannot be considered as a `must' but the usability of the corpus - especially for ASR - will increase considerable if one provide this information together with the corpus. On the other hand, the choice of how to encode the pronunciation of corpus can range from very simple and achievable with automatic procedures to very complex and time-consuming. In any case be prepared to reserve some man power for the creation of a dictionary, because in most cases there is manual work involved.

2.7.7 Validation

Basically every item included in the specifications of the speech corpus may be subject to validations. What will be validated and what will be regarded as an error has to be included in the contract or the corpus specifications. Typically, the following parts of the speech corpus are validated:

- Documentation: consistency, completeness, structure
- Meta data: completeness, parsability, contents (samples)
- Signal data: completeness, technical quality, acoustical quality (samples), contents (samples)
- Annotation data: completeness, parsability, contents (samples)
- Media: readability (on different computer platforms)
- Dictionary: completeness, quality (samples)

2.7.8 Documentation

Although not very common, the documentation procedures might also be specified before-hand. This is probably a good idea in large projects with several producing partners working on a common goal. Partners could use a distributed pre-formatted documentation template and fill it in accordingly. This might facilitate the validation of the documentation as well.

2.7.9 Distribution

The final stage of the speech corpus production is the production of distributable media. The major points to be considered here are:

- Which media to choose and how to produce them
- Compression and Compatibility problems
- Signal data vs. symbolic data
- Safety, verify procedures and version control
- Larger edition vs. Burn-on-Demand
- On-line distribution

2.8 Difficulties in building Speech Recognition Corpus

Large vocabulary continuous speech recognition Corpus is hard to build. That's why there are a few related researches works on Bangla corpus which are done in India but not in Bangladesh[8]. There are some specific issues behind this:

2.8.1 Speaker Adaptation

Speaker independence is highly desirable since it allows a system to be used straight out of the box and it allows systems to be built for which the speaker is not known in advance. However, in many applications, a speaker will either become a regular user or will input a reasonable quantity of speech at first use. In such cases it is natural to adapt the acoustic models to match that speaker. Adaptation can result in substantial reductions in error rate; particularly for atypical speakers and its incorporation in future LVR systems will be essential.

Adaptation can be supervised or unsupervised and it can be performed incrementally as the speaker is talking or offline at the end of the session. Each of these different styles may be most appropriate for some particular application. However, unsupervised incremental adaptation is the least intrusive and most generally useful to the user. Current research is therefore particularly concerned with techniques which can yield worthwhile performance gains with very little adaptation data but which also asymptotically lead to speaker dependent performance once a large amount of data has been acquired.

2.8.2 Environmental Robustness

Robustness to background noise and channel variability is clearly an essential requirement for the widespread use of LVR technology. As explained above, LVR systems utilize acoustic feature vectors consisting of short term spectral estimates to which some form of amplitude compression has been applied such as a log operation.

The primary effect of additive noise is to shift the spectral means and shrink the variances. However, it is important to note that the addition of noise changes the whole distribution not just the means and variances. Hence, a secondary effect of noise is to change the modeling assumptions. Channel variability is convolutive noise which after a log operation becomes a simple, but possibly time-varying, offset.

In the large vocabulary area, noise-robustness remains a substantially unsolved problem. As the results, without any form of compensation the performance of an LVR system will drop dramatically in noise. Compensation can limit this effect but cannot yet given immunity.

2.8.3 Task Independence

Whereas the acoustic models in an LVR system are relatively task-independent, the language model is typically trained on a large corpus of task specific material. This leads to systems which are inherently task dependent. For example, a language model trained on office correspondence will not work well on legal documents. Moving domain typically results in a lowering of accuracy due to the language model mismatch and an increase in the incidence of out-of-vocabulary (OOV) words.

In the long term, task dependence will be reduced through a general increase in language modeling capability. A recent trend in computational linguistics has been an increased interest in statistical techniques and this should eventually lead to the incorporation of explicit grammatical knowledge into LVR systems. In the nearer term, solutions being studied include multiple-domain modeling and adaptation. In the former, individual LMs are trained on a range of possible domains and then combined as a mixture model. Adaptive systems typically combine a static LM trained offline with a dynamic cache of N-grams which are being continuously updated.

2.8.4 Spontaneous Speech

Much of the recent research effort in the LVR area has been directed at transcribing read speech. However, the ability to transcribe spontaneous casual speech is also an important requirement and this is now receiving increased attention.

Current experimental evidence, particularly using the Switchboard Database, suggests that the error rate of current systems increases substantially when applied to spontaneous speech. The reasons for this are still unclear but probably stem from a number of factors including poor articulation, increased co-articulation, highly variable speaking rate and various types of dis-fluency such as hesitations, false-starts and corrections.

2.8.5 Real Time Operation

In addition to obtaining an acceptable level of recognition accuracy, computationally efficient implementation is needed in order to exploit LVR technology. As explained in section, recognition in LVR systems involves searching a very large space of hypotheses. In continuous density systems, the computational cost of this search will be divided between building and searching the network structure and evaluating Gaussian densities.

2.9 Speech Recognition in Machine

The Speech recognition that is performed in machine consists of three phases.

The stages are:

1. Speech Signal Analysis
2. Training
3. Recognition

2.9.1 Speech Signal Analysis

Applications that need voice processing require specific representations of speech information. For instance, the main requirement for speech recognition is the extraction of voice features which may distinguish different phonemes of a language. This procedure is equivalent to finding a sufficient statistics to estimate phonemes[12]. This portion consists of-

- Voice is a pressure wave **P(t**), which is converted into numerical values in order to be digitally processed, **X_c(t).**
- The "Mel Frequency Cepstral Coefficients" **(MFCC)** is find out by feature extraction by signal preprocessing, windowing, spectral analysis, filter bank processing and log energy computation.
- Then cepstrum analysis is done by DFT and IDFT.
- After that distortion measurement is done to end the speech signal analysis.

2.9.2 Training

The training process involves preparing a sample speech corpus that consists of a considerable number of sample spoken sentences. Each of this speech is input to the training module. The trainer then performs the labeling and segmentation operation on these input signals. This involves labeling the phonemes and word boundaries of the speech input signals[10]. After features like Mel Frequency Cepstral Coefficients are extracted from the audio input and then are represented in frequency vectors. These vectors are used to train the machine for patterns. These patterns can be phonemes or words. Using these features a model classifier is built to classify any pattern next time.

2.9.3 Recognition

In the recognition phase, the system is tested with a number of speech signals that are entirely different from the ones that were used to train the system. From these input the recognizer/decoder performs a feature analysis and extracts the features[16]. From there on using the patterns extracted, the model classifier identifies which phonemes matches these patterns and concatenate them to for individual words.

2.10 Speech Recognition Tools

There are a number of Speech Recognition tools that are available to use and build a recognition system for any language. Each of these alternatives were compared against for factors like availability, performance, documentation, supported platforms and languages[10]. The table below shows the comparison between the available tools

Table 2.1: Speech Recognition Tools Comparison

	License	Development Language	Platforms	Support
Julius	BSD	C	Linux/Unix, Windows	-
HTK	Prohibits redistribution and commercial use but R&D allowes	C	windows, Linux/Unix, Mac OS X	HTK book, Active mailing list
Sphinx	BSD	C, Java	windows,	Very well

			Linux/Unix, Mac OS X	documented
ISIP ASR	Public Domain (no restriction)	C++	Windows	-

2.10.1 Julius

Julius is an open source speech recognition engine. Julius is a high-performance, two-pass large vocabulary continuous speech recognition (LVCSR) decoder software for speech-related researchers and developers. It can perform almost real-time decoding on most current PCs in 60k word dictation task using word 3-gram and context-dependent HMM. Major search techniques are fully incorporated. It is also modularized carefully to be independent from model structures, and various HMM types are supported such as shared-state triphone and tied-mixture models, with any number of mixtures, states, or phones. Standard formats are adopted to cope with other free modeling toolkit. Julius has been developed as part of a free software toolkit for Japanese LVCSR research since 1997, and the work has been continued at Continuous Speech Recognition Consortium (CSRC), Japan from 2000 to 2003[13].

2.10.2 HTK

HTK (Hidden Markov Model Toolkit) is software toolkit for handling HMMs. It is mainly intended for speech recognition, but has been used in many other pattern recognition applications that employ HMMs, including speech synthesis, character recognition and DNA sequencing. Originally developed at the Machine Intelligence Laboratory (formerly known as the Speech Vision and Robotics Group) of the Cambridge University Engineering Department (CUED), HTK is now being widely used among researchers who are working on HMMs[8].

2.10.3 CMUSphinx

CMU Sphinx, also called Sphinx in short, is the general term to describe a group of speech recognition systems developed at Carnegie Mellon University. These include a series of speech recognizers (Sphinx 2 - 4) and an acoustic model trainer (Sphinx Train)[8].

2.10.4 ISIP ASR

The primary goal of our Internet-Accessible Speech Recognition Technology project is to create a freely available, modular, state-of-the-art speech recognition system that can be easily modified to suit your research needs. The system is built on top of a vast hierarchy of general purpose C++ classes that implement generic math, data structure, and signal processing concepts[8].

2.11 Reason to Choose Sphinx

By considering the factors mentioned above and few others Sphinx was chosen for this thesis. The reasons are:

- Runs on any platform

- Very well documented. Help is available from SourceForge group.

- Development language is C and Java both of which is very familiar

- BSD license available, so it can be used for free.

Chapter 3: Methodology

This Chapter describes the complete procedure of our implementation of speech recognition corpus and testing of it.

3.1: Required Tools:

We needed several softwares for our thesis

3.1.1 Audacity

Audacity is a free open source digital audio editor and recording computer software application, available for Windows, Mac OS X, Linux and other operating systems. Audacity was started in May 2000 by Dominic Mazzoni and Roger Dannenberg at Carnegie Mellon University. In addition to recording audio from multiple sources, Audacity can be used for post-processing of all types of audio, including podcasts by adding effects such as normalization, trimming, and fading in and out.[Audacity has also been used to record and mix entire albums, such as by Tune-Yards. We have used this tool for recording speeches in constant 16bit sample rate mono 10kHz sampling frequency sound[14].

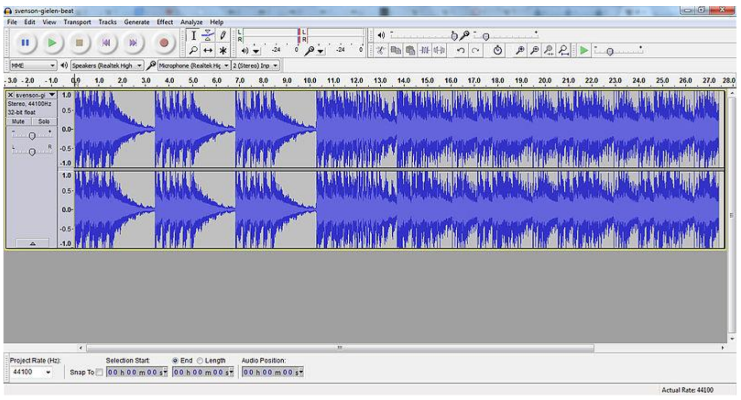


Figure 3.1: Audacity

3.1.2 Linux

Linux is a Unix-like and mostly POSIX-compliant computer operating system assembled under the model of free and open source software development and distribution. The defining component of Linux is the Linux kernel, an operating system kernel first released on 5 October 1991 by Linus Torvalds. it is used to install our software CMUSphinx because it operates well in linux operating system[15]. We used ***Linux: UBUNTU Backtrack KALI 14.04***

3.1.3 CMUCLMTK

CMUCLMTK is a part of CMUSphinx speech recognition system it is used to build language model that is needed to test the speech recognition system[19]. We used ***CMUCLMTK-0.4.***

3.1.4 SphinxTrain

SphinxTrain is also a part of CMUSphinx speech recognition system. It is used to train the HMM with a speech corpus. We use ***SphinxTrain Nightly.***

3.1.5 PocketSphinx

PocketSphinx is a speech recognizer developed by Carnegie Mellon University. It is also integrable to Android mobile phone. We used it to test our speech corpus. We used ***PocketSphinx-0.5***

3.1.6 SphinxBase

SphinxBase is the main part of CMUSphinx speech recognition system. It is required to install and use SphinxTrain and PocketSphinx. We used ***SphinxBase Nightly***.

3.1.7 Dos2Unix

dos2unix (sometimes named tounix or d2u) is a tool to convert line breaks in a text file from DOS format (Line feed+ carriage return) to UNIX format (Line feed) and vice versa. When invoked as dos2unix the program will convert a DOS text file to UNIX format. It is used as a terminal command when installed in Linux.

3.2 Building Bangla Speech Corpus

Speech corpus is the most important portion of speech recognition that provide the information to speech recognition system. So building a speech corpus is the most important task.

3.2.1 Text Preparation

The corpus speech is chosen from various resources. We used a book from Humayun Ahmed, a book from dr. Jafor Iqbal, speeches from Siraj-ud-doula, and some famous dialogs from movies and serials. A portion of Text that has been uttered is given below

```
jhum bristir shobde ghum bhanglo
 ai brishtir arek nam aula jhaula brishti
 kichukhon dokkhin dk theke fota porche kichukhon uttor dik
theke
 majhe majhe batasher jhapta
```

```
samanno biroti abar shuru
mesbarir ekta ongshe tiner chad
sekhane shila brishtir jhon jhon shobdo o holo
beparta ki
novembor mash brishti badlar mash
```

3.2.2 Sound Files

In order to train the ASR system, it is necessary to record sample audio files consisting of sentences that has been produced in text preparation section. In order to be an acoustic model to be accurate it is necessary that the system is trained using recordings of sentences of different speakers. This ensures that the system is trained on different utterances and hence be able to recognize more easily. In order to train the system we have built a speech corpus by recording 600 sentences in the voice of 10 different speakers, 8 of whom were male and the rest were female of the age range 20-25. And their personal profile includes information like

- Name
- Gender
- Language dialect

The recording parameters that were used are:

- Bit rate (bits per sample): 16 bits
- Sampling rate of the audio: 16 Khz
- Channel: Mono (Single Channel)

For this work 16 kHz sample rate has been chosen because it provides more accurate high frequency information and 16 bit per sample will divides the element position in to 65536 possible values.

After the recording, the splitting of the audio files per sentence has been done manually using Audacity and saved in a .wav format, where each wav file has been named by using department id, roll no. and sentence id.

For example: 07110004.wav stands as

Dept. Id: 07   Roll no: 11   Sentence Id: 0004

3.2.3 Transcription File

A transcript is needed to represent what the speakers are saying in the audio file. So in a file the dialogue of the speaker noted exactly the same precise way it has been recorded, with silence tag (starting tag <s> , ending tag </s>), saved by the file id which represent the utterance. The transcription file is named as

<sound_file_name_ID>.transcript

Example:

```
<s> jhum bristir shobde ghum bhanglo </s>
<s> ai brishtir arek nam aula jhaula brishti </s>
<s> kichukhon dokkhin dk theke fota porche kichukhon uttor dik
theke </s>
<s> majhe majhe batasher jhapta </s>
<s> samanno biroti abar shuru </s>
<s> mesbarir ekta ongshe tiner chad </s>
<s> sekhane shila brishtir jhon jhon shobdo o holo </s>
<s> beparta ki </s>
<s> novembor mash brishti badlar mash </s>
```

3.2.4 Label File

This file contains the phonetic representation of the uttered sentences. It maps to a sequence of sound units to derive the sequence of sound units associated with each signal. To make this file, resources from CRBLP pronunciation dictionary is used. In this case the output of the pronunciation sound unit is given in Unicode IPA but for training a system the dictionary files need sound units in ASCII. For this reason a program is made which can translate IPA Unicode to ASCII. After all this, the files are saved with .phone extension.

For example, for the sentence “*majhe majhe batasher jhapta*” The phone file contains

```
sil
m
aa
Jh
e
m
aa
Jh
e
b
aa
T
aa
sh
e
r
Jh
aa
p
T
```

```
aa
sil
```

A sample wave for a word with phonetic representation is

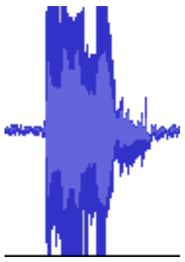
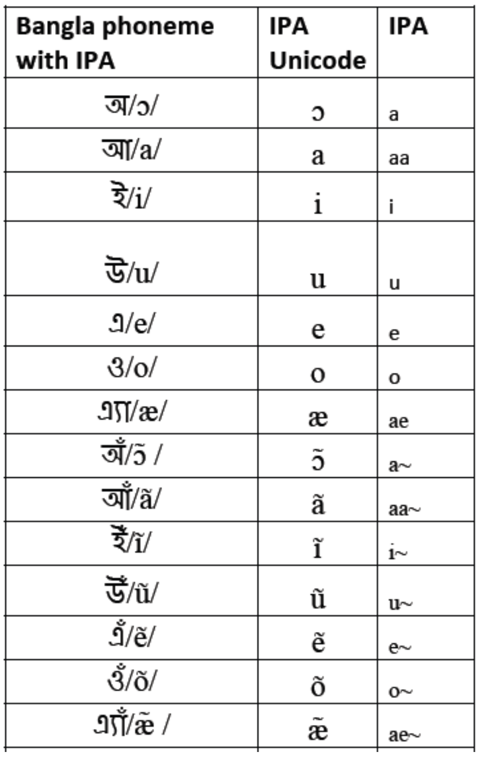


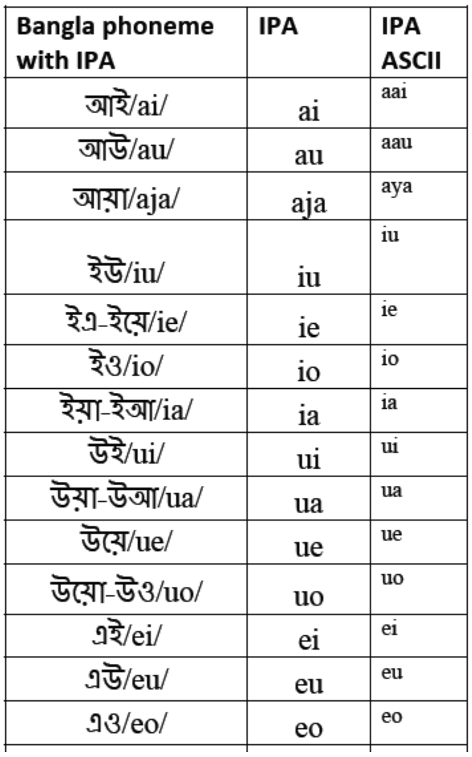
Figure 3.2: Wave form of "AMAR" (aa m aa r)

3.2.4.1 IPA Unicode to ASCII Chart

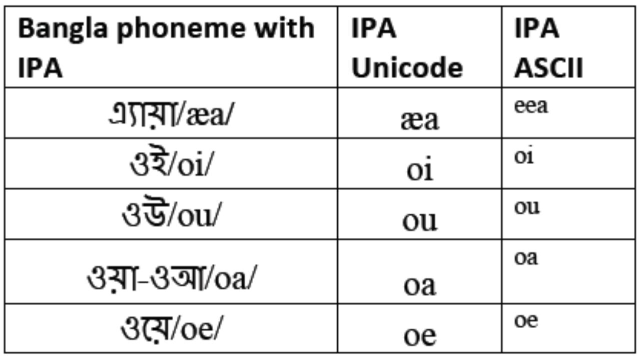
Bangla phoneme with IPA	IPA Unicode	IPA ASCII
ক /k/	k	K
খ /k^h/	k^h	Kh
গ /g/	g	G
ঘ /g^h/	g^h	Gh
চ /c/	c	C
ছ /c^h/	c^h	Ch
য,জ /ɟ/	ɟ	J
ঝ /ɟ^h/	ɟ^h	Jh
ট /t/	t	T
ঠ /t^h/	t^h	Th
ড /d/	d	D
ঢ /d^h/	d^h	Dh
ত /t̪/	t̪	Ta
থ /t̪^{h}/	t̪^{h}	Tah
দ /d̪/	d̪	Da
ধ /d̪^{h}/	d̪^{h}	dah
প /p/	p	p
ফ /p^h/	p^h	ph
ব /b/	b	b
ভ /b^h/	b^h	bh
শ,ষ,স/ʃ/	ʃ	sh
শ,স/s/	s	s
ম /m/	m	m
ঙ,ং /ŋ/	ŋ	ng
ণ, ন /n/	n	n
র /r/	r	r
ল /l/	l	l
ড়, ঢ় /ɾ/	ɾ	ra
য় /j/	j	y



Bangla phoneme with IPA	IPA Unicode	IPA
অ/ɔ/	ɔ	a
আ/a/	a	aa
ই/i/	i	i
উ/u/	u	u
এ/e/	e	e
ও/o/	o	o
এ্যা/æ/	æ	ae
অঁ/ɔ̃ /	ɔ̃	a~
আঁ/ã/	ã	aa~
ইঁ/ĩ/	ĩ	i~
উঁ/ũ/	ũ	u~
এঁ/ẽ/	ẽ	e~
ওঁ/õ/	õ	o~
এ্যাঁ/æ̃ /	æ̃	ae~



Bangla phoneme with IPA	IPA	IPA ASCII
আই/ai/	ai	aai
আউ/au/	au	aau
আয়া/aja/	aja	aya
ইউ/iu/	iu	iu
ইএ-ইয়ে/ie/	ie	ie
ইও/io/	io	io
ইয়া-ইআ/ia/	ia	ia
উই/ui/	ui	ui
উয়া-উআ/ua/	ua	ua
উয়ে/ue/	ue	ue
উয়ো-উও/uo/	uo	uo
এই/ei/	ei	ei
এউ/eu/	eu	eu
এও/eo/	eo	eo



Bangla phoneme with IPA	IPA Unicode	IPA ASCII
এ্যায়া/æa/	æa	eea
ওই/oi/	oi	oi
ওউ/ou/	ou	ou
ওয়া-ওআ/oa/	oa	oa
ওয়ে/oe/	oe	oe

3.2.5 Directory File

It is just a plain text file containing all the name of each of the audio file without extension. This file is saved with .dir format

For example:

07110001 07110002 07110003 …..

3.2.6 Phone File

A simple text file that tells a trainer what phonemes are part of the training set. The file has one phone in each line, no duplicity is allowed. This file was generated manually. It is named as *fonemi.dat.*

For example,

iu

ie

io

ia

…

3.2.7 Dictionary File

This file contains the words that is used in corpus and its phonetic representation separated by a space and no of phones used for utterance of it. The words are in soted order and one word per line. it is named as *vocabula.dat.*

Example

```
Word Phoneme   No. of phoneme
abar /aa/b/aa/r     4
ache /aa/Ch/e  3
ai   /aai 1
akashe    /aa/K/aa/sh/e  5
amader    /aa/m/aa/D/e/r 6
ar   /aa/r     2
arek /aa/r/e/K 4
```

```
ashar     /aa/sh/aa/r    4
aula /aau/l/aa 3
badlar    /b/aa/D/l/aa/r 6
```

3.2.8 Word File

This file contains the words used in the corpus in sorted order. It is named as *words.dat*

Example

```
bhanglo
bhengeche
bhongite
bichanai
biroti
borof
boshe
boshtei
brishti
brishtir
bristir
carbon
chad
chere
choumbok
dangai
dear
di
```

3.2.9 Occurrence File

This file keeps the record of how many time a phoneme is uttered in the speech corpus. This file is saved as *no_of_occ.tra.* This is created with a short C program.

3.2.10 Transition File

This file keeps the record of how many times a phoneme is uttered after a particular phoneme. This file is saved as *no_of_trans.tra.* This is created with a short C program.

3.2.11 Bigram file

Bigram model is an important part of language modeling. It is computed using

$$P(W) = P(Wi)\prod_{i=2}^{N} P(Wi|Wi-1)$$

where *P(WilWi-1)* is the probability of a phoneme with repect to its previous phoneme. It is generated by C programming and saved as *bi_gram.tra.*

3.3 Checking Bangla Speech Corpus Using CMUSphinx

We need to check our Speech corpus for determining the performance. We have done it by PocketSphinx.

3.3.1 Setting up the System

3.3.1.1 Prerequisite

To setup CMUSphinx, Perl and C compiler is needed. Perl will be useful to run the scripts where as to compile the source code a C compiler is needed[19]. This project has been entirely setup in Linux platform, so to check whether both Perl and C compiler are present, in the terminal:

For **Perl** type:

```
$ dpkg -s perl
```

If it is installed in the system then it will show all the details else to download and install Perl type in the terminal:

```
$ sudo apt-get install perl
```

For **C compiler** type:

```
$ dpkg -s gcc
```

If gcc is already installed then related information will be shown else download it by typing in terminal:

```
$ sudo apt-get install gcc
```

3.3.1.2 Setting up the Recognizer Software

To process the speech corpus the following steps are involved-

1. A directory has been created for the system, and moved to that directory. This is the main directory of the system.

```
mkdir sprinx
cd sprinx
```

2. The speech corpus has been copied in this directory.

3. The SphinxTrain zip file has been copied to the directory and unzipped by-

```
$ gunzip -c SphinxTrain.nightly.tar.gz | tar xf -
```

4. Then we have navigated to the SphinxTrain folder and install it by

```
$ cd SphinxTrain
$ ./configure
$ make
```

5. After that the Sphinxbase zip file has been copied to the first directory,unzipped it, navigated to the folder and installed by following commands

```
$ gunzip -c sphinxbase.nightly.tar.gz | tar xf -
$ cd sphinxbase
$ ./configure
$ make
```

6. Again the decoder Pocketsphinx zip file has been copied to the main directory, unzipped it, navigated to the folder and installed by following commands

```
$ tar zxf pocketsphinx-0.5-20080912.tar.gz
$ cd pocketsphinx
$./build_all_static.sh
```

7. As the whole Speech corpus is created in windows operating system, some text files will not work in linux OS. So they are converted to linux-supported text by going to the *backups* folder of speech corpus, installing and using Dos2Unix.

```
find . -type f -exec dos2unix {} \
```

Now the condition of the main directory is

```
sprinx
  • Bangla Speech Corpus
  • pocketsphinx
  • pocketsphinx-0.5-20080912.tar.gz
  • SphinxTrain
  • SphinxTrain.nightly.tar.gz
  • sphinxbase
  • sphinxbase.nightly.tar.gz
```

3.3.1.2 Building Language Model for CMUSphinx

CMU sphinx requires a special dump(.dmp) file as language model. This is done using CMUCLMTK guide[18]. The process is

1. A file named *input.txt* is created using a small C program cumulating some *.transcript files.

2. Then a command promt has been opened and navigated to CMUCLMTK folder.

3. This command has turn *input.txt* into a word frequency list

```
text2wfreq.exe <input.txt >input.wfreq
```

.

4. This command has turned the word frequency list into a vocabulary list

```
wfreq2vocab.exe <input.wfreq >input.vocab
```

.

5. This has created an idngram file called input.idngram

```
text2idngram.exe -vocab input.vocab -idngram input.idngram
<input.txt
```

6. This command create a .arpa formatted file.

```
idngram2lm.exe -idngram input.idngram -vocab input.vocab -arpa input.arpa
```

7. Then we have navigated to *Sphingbase/bin/debug* folder and run this command to create language model dump file.

```
sphinx_lm_convert -i input.arpa -o uccharon.lm.DMP
```

3.3.1.3 Setting up the corpus

The speech corpus has to be set up for the speech recognition system with SphinxTrainer[20]. The process is-

1. First a folder for the project (this project is named as ***Uccharon***) has been created, which is in the main directory. Then navigated to the project folder and there we run the below command in the terminal

```
$ perl '../SphinxTrain/scripts_pl/setup_SphinxTrain.pl' -task
Uccharon
```

2. After this in the project folder, these files has been created.

```
etc
feat
logdir
model_parameters
model_architecture
wav
```

All the folders except wav and etc will be fulfilled during training

3. In wav folder, all the audio files (*.wav) has been copied.

4. Then in etc folder all the prepared files prepared from our corpus have been copied so the folder looks like:

```
Uccharon/etc:
  ▪ Uccharon.dic (same as vocabula.dat)
  ▪ Uccharon.filler (a special file created for silence
    detection)
  ▪ Uccharon.phone (same as fonemi.dat)
  ▪ Uccharon.fileids (portion of sndfiles.dir)
  ▪ Uccharon.transcription (Collection of *.transcript files)
  ▪ feat.params [created automatically]
  ▪ sphinx_train.cfg [created automatically]
```

There must not be any extra line at the end of all the files mentioned above.

5. Few changes has been bought in the configuration file for further proceeding. The changes are:

```
  ➢ The extension of the wavfiles has been changed from 'sph'
```

to 'wav' i.e $CFG_WAVFILE_EXTENSION = 'wav'

- There are few common sound file formats, some of them are nist, raw, mswav etc. In our cfg file the wav type has been altered to 'raw' from 'nist' $CFG_WAVFILE_TYPE = 'raw'
- States per HMM will be 3
- $CFG_FEATURE will be "1s_c_d_dd"
- $CFG_FINAL_NUM_DENSITIES = 1
- $CFG_N_TIED_STATES = 100

6. The next step is to convert the raw speech data from the wav files to MFCC (Mel Frequency Cepstral Coefficients)and store them in Uccharon/feat directory. For this we move to the project folder, Uccharon and run the below command:

```
$ perl scripts_pl/make_feats.pl -ctl etc/Uccharon.fileids
```

7. Now we have run the following command to check if the corpus is ready to go

```
$ perl scripts_pl\00.verify\verify_all.pl
```

and have gotten this output that indicates it is ready to train.

```
MODULE: 00 verify training files
O.S. is case insensitive ("A" == "a").
Phones will be treated as case insensitive.
    Phase 1: DICT - Checking to see if the dict and filler dict
agrees with the phonelist file.
        Found 1219 words using 40 phones
    Phase 2: DICT - Checking to make sure there are not
duplicate entries in the dictionary
    Phase 3: CTL - Check general format; utterance length (must
be positive); files exist
```

```
    Phase 4: CTL - Checking number of lines in the transcript
should match lines in control file
    Phase 5: CTL - Determine amount of training data, see if
n_tied_states seems reasonable.
        Total Hours Training: 1.52594188034191
        This is a small amount of data, no comment at this time
    Phase 6: TRANSCRIPT - Checking that all the words in the
transcript are in the dictionary
        Words in dictionary: 1216
        Words in filler dictionary: 3
    Phase 7: TRANSCRIPT - Checking that all the phones in the
transcript are in the phonelist, and all phones in the phonelist
appear at least once
```

3.3.2 Training

After all system setup and data preparation is done, the next step is to train the system and to train the system, most of the audio files from all speaker has been used as data. And for training the system scripts in scripts_pl directory will needed to be executed.

There are two option, one of them is to run:

```
$ perl scripts_pl/RunAll.pl
```

As this scripts will execute all the available perl scripts. Another option is run only required scripts one by one in the sequence given below.

```
1.perl scripts_pl/00.verify/verify_all.pl
```

This scripts verify whether all the files are in correct format, checks if there is any mismatch in transcription file, phone file and dictionary files, detects extra space etc.

```
2. perl scripts_pl/01.vector_quantize/slave.VQ.pl
```

This script is only needed to execute if the training is given for continuous model

```
3. perl scripts_pl/20.ci_hmm/slave_convg.pl
```

This script can give error messages if the dictionary have any duplicate entries but while data preparation this problem should be taken care.

```
4. perl scripts_pl/30.cd_hmm_untied/slave_convg.pl
```

```
5. perl scripts_pl/40.buildtrees/slave.treebuilder.pl
```

As described in data preparation section, that in phone file there should not be any duplicate entries and no extra phone which is not in the dictionary should be allowed but if there is then warning message like

```
6. perl scripts_pl/45.prunetree/slave.state-tying.pl
```

```
7. perl scripts_pl/50.cd_hmm_tied/slave_convg.pl
```

```
8.perl
scripts_pl/90.deleted_interpolation/deleted_interpolation.pl
```

```
9.perl scripts_pl/99.make_s2_models/make_s2_models.pl
```

During the training process several new director has been created which contains files that are going to be included in the acoustic model.

3.3.3 Recognition using Pocketsphinx

The recognition using pocketsphinx involves-

1. To setup decoder navigate to the task folder (Uccharon) and type

```
$ perl ../pocketsphinx0.5/pocketsphinx/scripts/setup_sphinx.pl -
task Uccharon
```

2. Then this files are added in

```
Uccharon/etc
   o  Uccharon _test.transcription
   o  Uccharon _test.fileids
   o  >sphinx_decode.cfg [generated during PocketSphinx setup]
```

3. Then convert .wav files to .mfc files using command:

```
$ perl scripts_pl/make_feats.pl -ctl etc/Uccharon_test.fileids
```

4. Before starting to decode the test files a modification is required to make. We had to add

```
        -dictcase => "yes"

  to scripts_pl/decode/psdecode.pl
```

5. Then to decode type

```
$perl scripts_pl/decode/slave.pl
```

6. Then after the process is completed we have been getting an output like and saved in "Uccharon.align":

```
Decoding 529 segments starting at 0 (part 1 of 1) 0%

This step had 2 ERROR messages and 45 WARNING messages. Please
check the log file for details.

Aligning results to find error rate

SENTENCE ERROR: 2.8% (15/529) WORD ERROR RATE: 1.0% (23/2319)
```

Chapter 4: Result and Analysis

4.1 Result

ASR performance reflects the quality of speech corpus. We have followed the corpus approach described in Claudio *Becchetti*'s book. We trained the CMUSPHINX with 400 sentences. Then we tested with other 40 speeches. Different test set results are described below-

Experiment no.	Details	Result
1	Trained with: 40 speech Tested with: 5 speech	45.21%
2	Trained with: 400 speech Tested with: 10 speech	40.11 %
3	Trained with: 400 speech Tested with: 20 speech	36.21%
4	Trained with: 400 speech Tested with: 30 speech	34.11 %

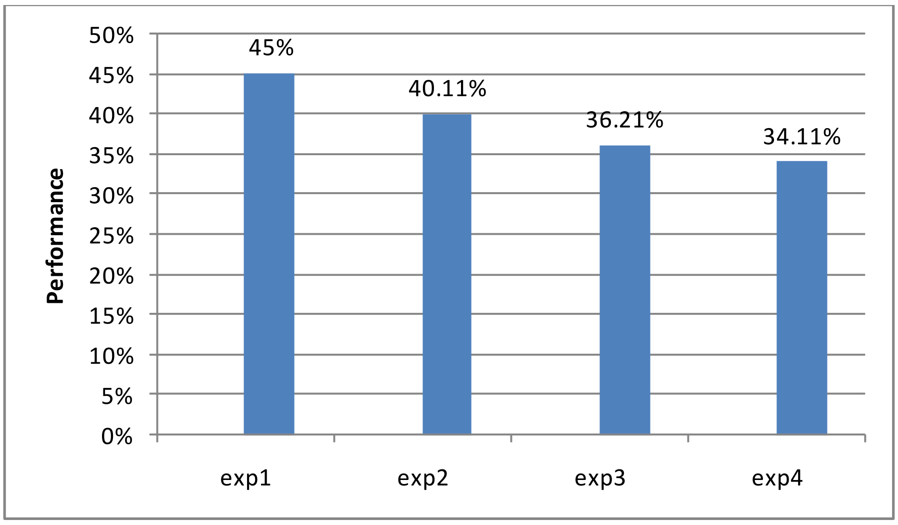


Figure 4.1: Performance of Corpus

4.2 Analysis

We have built a speech corpus and determine its performance measure is approximately **39.39 %.** No speech recognizer has gained 100% accuracy. We have analyzed the result and come to some proposals to make-

- Bangla is a very rich language. It has 11 vowels and 39 consonants. So it should not be compared to English language phonetic analysis. It is best if our own Speech recognizer can be made where the number of iteration in Baum-welch parameter re-estimation method and forward chaining can be very large.
- We used normal phoneme structure. It would better if PRAGMA phoneme structure can be used. We assume "j" and "J" are same. But in PRAGMA, they are different.
- And of course, the more you train, the more accurate result can be achieved. So enhancing a speech corpus with more accurate training, adaptation of system with environment, speaker adaptation can give us enough accuracy.

Chapter 5: Conclusion

In this thesis paper, we build a continuous read speech corpus. We have developed speech corpus of standard Bangla language which is mostly spoken in Bangladesh. ASR performance degrades due to speaker variability, environment noise and transmission channel noise. Phoneme recognition with monophonic model and phone model gives good accuracy. We have used 13 base MFCC feature and its delta and delta-delta feature.

We have developed also an automatic recognition system with CMU SPHINX. This speech corpus can be further used for future development of Bangla Automatic speech recognition system. It can be a great resource for Bangla speech to text applications also. We hope to see a complete ASR system for Bangla language of our own.

Reference

[1] Speech Recognition- Wikipedia, the free encyclopedia Link: http://en.wikipedia.org/wiki/Speech_recognition

[2] Lawrence R. Rabiner, “A Tutorial on Hidden Markov Models and selected applications in Speech Recognitions,”

[3] Ir. p. Wiggers, Dr. drs. L.J.M. Rothkrantz, “Automatic Speech Recognition using Hidden Markov Model,”

[4] Mikael Nilson, “Speech Recognition using Hidden Markov Model performance evaluation in noisy environment,”

[5] BKM Mizanur Rahman, Bulbul ahmed, “Gender Effect canonization for Bangla ASR System,”

[6] Speech Recognition System and C++ implementation by C. Becchetti and R.P Richotti

[7] Automatic Speech recognition System. Link: www.speech.bme.ogi.edu

[8] Speech Corpus Production. Link: http://www.bas.uni-muenchen.de/forschung/BITS/TP1/Cookbook/node37.html

[9] “Bangla Large Vocabulary Continuous Speech Recognition with Hidden Markov Model Toolkit” – Thesis Paper by Ashiqur Rahman and Abidur Rahman Mallik.

[10] “Speech Recognition System for Bangla”- Thesis Paper(BRACCU) by Shammur Afsar Chowdhury

[11] “Development of isolated speech recognition system for Bangla words” by Mizanur Rahman and Fatema Khatun.

[12] “Comparative Study of Feature Extraction Methods for Bangla Phoneme Recognition” by F. Khan and R.C. Debnath

[13]Julius- Wikipedia. Link: http://en.wikipedia.org/wiki/Julius_%28software%29

[14]Audacity- Wikipedia. Link: http://en.wikipedia.org/wiki/Audacity

[15]Linux- Wikipedia. Link: http://en.wikipedia.org/wiki/Linux

[16] “Algerian Arabic speech database (ALGASD): corpus design and automatic speech recognition application” by M.H. Hamdani

[17] “Speech Corpus for Continuous Automatic Speech Recognition” by S. Roy

[18]Ming Xiao’s Blog. Link: http://musingsfromming.blogspot.com/2012/05/creating-language-model-with-cmuclmtk.html

[19]CMUSPHINX Homepage. Link: http://cmusphinx.sourceforge.net/

[20]Robust Group Tutorial. Link: http://www.speech.cs.cmu.edu/sphinx/tutorial.html#sphinx3tarball

[21] “The CMU Arctic speech databases” by John Kominek, Alan W Black