ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF INFORMATICS

INFORMATION SCIENCE DEPARTMENT

Hidden Markov Model Based Large Vocabulary, Speaker Independent, Continuous Amharic Speech Recognition

A Thesis Submitted to the School of Graduate Studies of Addis Ababa University in Partial Fulfillment of the Requirements for the Degree of Masters of Science in Information Science

BY

ZEGAYE SEIFU

JUNE 2003

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION STUDIES FOR AFRICA

Hidden Markov Model Based Large Vocabulary, Speaker Independent, Continuous Amharic Speech Recognition

BY

ZEGAYE SEIFU WUBISHET

Signature of the Board of Examiners for Approval

_____	_____
_____	_____
_____	_____
_____	_____

DEDICATION

This thesis is dedicated to all people who have had their own share of contribution through out my life.

ACKNOWLEDGEMENTS

First and for most I would like to thank my advisors Ato Tesfaye Biru, Ato Solomon Birhanu and Ato Knfe Taddesse for their constructive comments and encouragement through out this work.

This research couldn't have come into being without the help of Solomon Tefera. Sol, I thank you for the corpus that you have given me. I thank you for your immediate response in helping me out whenever I got in trouble. I really thank you soul.

I am also grateful to my family who has rendered me all their care, love and encouragement. I am every thing today because yesterday they were at my back.

Finally, I would like to extend my heart-felt gratitude to all the staff members of the faculty. I thank them all for their kindness and support in every way possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF APPENDICES

ACRONYMS

ASR – Automatic Speech Recognition

HMM – Hidden Markov Model

HTK – Hidden Markov Model Toolkit

LPC – Linear Predictive Coding

MFCC – Mel Frequency Cepstral Coefficient

Abstract

This study investigated the possibility of developing large vocabulary, speaker independent and continuous speech recognizer for Amharic language. The recognizer was developed using Hidden Markov Model; and the Hidden Markov Modeling Toolkit was used to implement it.

In the process, a corpus developed by Solomon Tefera was used to get the required data for training and testing the models. It was a database comprised of 8000 utterances that were used for training and 500 plus sentences for development and evaluation. The data was preprocessed in line with the requirements of the HTK toolkit. In order to support the acoustic models, a bigram language model was constructed. In addition, pronunciation dictionary was prepared and used as an input.

Since the recognizer was meant to recognize large vocabulary and continuous speech, phonemes were opted as the basic unit of recognition. However phonemes are known to be context independent units, given that the environment in which a sound is put makes a difference in the way it is pronounced. Thus after the monophone based speech recognizer was built, it was promoted to triphone based system in which the left and right contexts were considered and modeled. Besides, the mixture components of the states of the models were incremented in view of optimizing the performance of the recognizers.

Performance tests were then conducted at various stages using the development and finally using evaluation test data. In the end, a 79% word level correctness, 76.18% word accuracy, and 30.01% sentence level correctness were obtained. The results are encouraging and with more optimization works better results can be achieved. Finally, conclusions were drawn and recommendations were made in line with the analysis and findings

Chapter One

Introduction

1.1.Background

1.1.1.What is Automatic Speech Recognition (ASR) – Overview

Speech is the ultimate, universal mode of human communication and it is how man should be able to interact with computers. Speech interface in user's own language, is in short, an ideal means of communication as being the most natural, flexible, efficient and convenient option allowing hands and eyes to be free (Nahm and Slater, 1997).

Literature reveals that there are at least two main areas of research in relation to man-machine communication by voice: speech synthesis (voice output) and speech recognition (voice input).

Speech synthesis is a technology for generating voice output from a computer. It converts a text to speech and read for the user (Zue and Cole, 1995). Computer based speech recognition on the other hand, is defined as the ability to take speech as input and produce a transcript of that speech as output. In technical terms, it is the conversion of an acoustic signal to a stream of words (Rodman, 1999).

Since the time of Alexander Graham Bell, one of the first people to take a serious look at developing ASR, speech recognition has undergone significant progress resulting in a number of practical applications, which have been successfully developed and used in a variety of areas. Over the last five decades in particular, spoken language interface to computers has been one area that lured and fascinated many engineers and speech scientists alike.

1.1.2. Approaches in developing Speech Recognition systems

In general it can be said that there are four basic approaches (though various writers may use different categorization) for developing speech recognition systems – acoustic-phonetic, template matching (pattern recognition), stochastic approaches, and Artificial Intelligence.

Acoustic-phonetic is the oldest, most straightforward and thoroughly researched method, whose principles are still used in the AI based recognizers .The approach assumes that the rules governing the phoneme variability are relatively simple and easily learnable.

> *It is based on the theory of acoustic phonetics that postulates that there exists finite, distinctive phonetic units in spoken language and that the phonetic units are broadly characterized by a set of properties that are manifested in the speech signal or its spectrum over time. Even though the acoustic properties of phonetic units are highly variable, both with speakers and with neighboring phonetic units (the so called co-articulation of sounds), it is assumed that the rules governing the variability are*

straightforward and can readily be learned and applied in practical situation (Rabiner and Juang, 1993).

This approach has limitations like absence of well-defined automatic procedure and standard linguistic way of labeling the training speech. Due to these, the acoustic – phonetic based speech recognition is no longer considered as the most interesting approach (Reiderer, 1999).

Template matching is one form of pattern recognition that represents speech data as sets of feature vectors called templates. Each word or phrase in an application is stored as a separate template. Spoken input by end users is organized into templates prior to performing the recognition process. The input is then compared with stored templates. Template matching is performed at the word level and contains no reference to the phonemes with in the word[1].

The stochastic approach entails the use of probabilistic models to deal with uncertain and incomplete information found in speech recognition. The term stochastic refers to the process of making a sequence of non-deterministic selections from among sets of alternatives. They are non-deterministic because the choices during the recognition process are governed by the characteristics of the input and not specified in advance. Both

---

[1] Simple pattern matching techniniques are not suitable for continuous speech recogniton because segment boundaries are difficult to detect (Odell, 1995).

Since a ixed set of example templates are used, this approach is best suited to small vocabulary speaker dependent recogniton (Ibid).

template matching and this approach require the creation and storage of models of each item to be recognized. However, stochastic processing involves no direct matching between stored models and input. Instead it is based upon complex statistical and probabilistic analyses that are best understood by examining the network like structure in which those statistics are stored. HMM, one type of stochastic process model is the basis of the majority of current speech recognition systems (Odell, 1995) and is employed in this work to construct Amharic continuous speech recognizer.

The main idea of the AI approach is to collect and employ knowledge from a number of sources in order to solve a problem in question. The knowledge sources are wide ranging from the fields of acoustic, lexical, syntactic, semantic and pragmatic knowledge (Rabner and Juang , 1993).

Most important techniques in the AI approach are the use of an expert system for segmentation and labeling of the acoustic signal; learning and adaptation over time; and the use of artificial neural networks (ANNs) or distinguishing between similar sound classes and learning the relations between all known inputs and phonetic data. The neural networks can represent a separate structural approach to speech recognition or can be regarded as an implementation architecture possibly incorporated in any of the three classical speech recognition approaches.

1.2. Statement of the Problem and Justification

It is no dubious that human beings from the early days on, have gone through various social and technological developments. The development of machines/technologies to the services of man in particular, has been growing so high and fast that it is sometimes difficult to believe. Apart from the design and use of the serving machines, the endeavor to tame these non-living machines and let them act like human is incredibly bearing fruit. One aspect of such attempts is letting machines recognize human speech in such away that they will be able to understand and respond to a human friend.

Such endeavors and attempts are never without reasons. Speech is the most natural means of communication between humans - one of the first skills we learn to use from our babyhood and which can be done without any tools or any explicit education. It is obvious that before the invention of writing, speech was the only way of passing knowledge. Even in the modern world, despite all our novel ways of communicating, the speak-listen style is opted most.

It is also the most important way of communicating. When speaking with somebody, one does not have to focus on the audience; he/she can look in a different direction or even perform some other task while communicating. So it is only logical that machine interface designers in their quest for a natural man-machine interface have turned to automatic speech recognition and speech production as one of the most promising interfaces

(Wiggers, 2001).

Rabiner and Juang (1993) stated that speech recognition is a rich field for both practical and intellectual reasons. As a practical area, it solves problems, improve productivity, and change the way we run our lives. Intellectually, it holds considerable promise as well as challenges in the years to come for scientists and product developers alike. From the practical point of view, speech interfacing serves a lot more problems of human beings, speedup the way people lead their day to day life and in turn help maximize productivity. Above all, speech is the most natural and simple way of communication mechanism ever. One can imagine how it feels like to have a friendly machine that lends wide-open ear, patient, amusing and serving.

Indicating the need for ASR, Zue (1987) once commented that:

> *Our long-term research goal is to develop a graceful environment for human-machine interaction. Because computers continue to play an ever increasing role in our lives, it is essential to create a habitable environment so that relatively unsophisticated users can take advantage of the available computing power We believe that interaction between a human operator and a computer can be made more natural by enabling the computer to communicate with humans via oral speech as an alternative to text.*

Given this reality, however, the speech recognition systems that have been developed so far are meant to serve a specific language and are totally limited to some techno-rich countries of the world. For developing countries like Ethiopia, it is only mandatory

to follow suit of those developed nations in relation to such technological advancements; or else they will be bound to stay at the bottom of progress because they fail to exploit to the best that the opportunities offered by technologies.

Based on this fact, speech engineers and language experts in various countries are making noticeable efforts to develop recognition that works for their own language. It is, therefore, necessary that similar endeavors be made on Ethiopian languages.

One of the most widely spoken languages in Ethiopia and the working language of the federal government is Amharic. Furthermore, Bender et.al. (1976) stated that, it is the most learned second language of all local languages. However, little has been done so far on the language in relation to speech technology. It is only in the past two years that glimmers of such works have started to sparkle. The works by Solomon (2001) and Kinfe (2002) are the only mentionable deeds and that actually light a candle in the area.

Solomon (2001) has tried to indicate the possibility of isolated Amharic consonant-vowel (CV) syllable recognition in his work. Kinfe (2002) has moved a step forward; he examined and compared the potential of three basic units of recognition – phoneme, triphone and CV syllable He did also indicate the possibility of discrete sub-word based Amharic word recognition.

The natural extension of these attempts and what actually remains to be done in consummating these endeavors towards Amharic ASR, was the development and realization of continuous Amharic speech recognition. The Holy Grail of any speech recognition system/study is the building of a speaker independent system that recognizes in real-time, natural, fluent, spontaneous, continuous speech and which is capable of telling speech and background noise apart. However, the current generation of systems, even in the developed languages, is nowhere near this ideal.

Therefore, this study aims at investigating and proving out the possibility of developing large vocabulary, speaker independent continuous Amharic speech recognition system using HMM. This will consummate the endeavor towards the construction of continuous Amharic speech recognition and the result will be an input for developments of real applications.

1.3. OBJECTIVES OF THE STUDY

General Objective

The general objective of the study was to investigate and demonstrate the possibility of developing a large vocabulary speaker-independent continuous Amharic speech recognizer.

Specific Objectives

To accomplish the above general objective, the following specific objectives were targeted

- Investigating the possibility of handling continuously spoken Amharic speech in view of realizing automatic continuous speech recognizer.
- Examining the effect of inter-speaker and intra-speaker variabilities and ways of handling them.
- Exploring and understanding how the Hidden Markov Model Toolkit (HTK) works and how it handles Amharic speech.
- Conducting a literature review on Automatic Speech Recognition in general and on how to develop a recognizer for Amharic Language.
- Building a prototype speaker-independent large vocabulary continuous speech recognizer using Hidden MarKov Model (HMM).
- Evaluating/testing the performance of the prototype recognizer.

1.4. METHODOLOGY

1.4.1. Review of Related Literature

Exhaustive literature review was performed to investigate the underlying principles/theories of the various approaches, techniques and tools that were employed in the research. In addition, literature on the Amharic language, especially those dealing with the phonetic/triphone features were reviewed. Moreover, to learn what others have done in the area and to better understand the problem, a comprehensive investigation of available empirical literature on automatic speech recognition was conducted.

1.4.2. Data selection and Preparation Methods

In order to build a robust large vocabulary speaker independent speech recognizer, it is crucial that all acoustic (sub-word) models receive enough training examples, taking into account the many variations that can occur in speech. It follows that a training corpus should be sufficiently large, consisting of speech samples spoken by men and women from different age and if possible linguistic background.

Since the sound of a sub-word unit is also influenced by its context the speech data should include all phonemes in as many different phonetic contexts as possible. In practice this means that, depending on the quality of the data and the complexity of the acoustic models, about ten- to thirty-thousand (phonetically rich) sentences are necessary (Wiggers, 2001).

Recording such data set obviously is a major undertaking involving sub-tasks like selection of phonetically rich and phonetically balanced sentences, selection of appropriate participants, recording the data and the most time-consuming parts post-processing and transcribing the data.

Studies of this kind in the developed languages are fortunate enough that many of these speech corpora are commercially available. However no such commercial database is there for Amharic. Despite this reality, Solomon Tefera, a researcher who is currently working his PHD on speech recognition, spared of this study. He has developed a corpus from 80 people with the required variety and each speaker having read 100 sentences, summing up to a total of around 8000 sentences. This actually fulfills the requirements of the study, which otherwise could have been a research undertaking by itself. In addition, the database contained development, evaluation, and adaptation data recorded from 20 people.

All the training data were used to develop the models in this work as such recognizers demands huge training data. For development test purposes, 18 speeches from 15 speakers (270 sentences) were used. For evaluation again 18 speeches of 15 speakers (270 sentences) were selected.

Apart from this a test data from 5 people, 4 male and 1 female, each having read 10 sentences, was recorded. These 50 sentences were needed to test the recognizer with a data recorded in a different environment from the training data. The recording was done in a laboratory environment, using Pentium IV /128 MB RAM /1.7 GH speed /and 40 GB HD multimedia desktop computer. The appropriate tool in Hidden Markov Model toolkit (HTK) was used to record the data.

The training, evaluation, development test and the adaptation data were recorded using a multimedia laptop computer - Pentium IV / 256 MB RAM /2.6 GH speed /40 GB HD.

1.4.3. Modeling Technique

The recognizer in this study was built using the popular Hidden Markov Model (HMM). HMM is a powerful technique capable of robust modeling of speech. It is a parametric model that is particularly suitable for describing speech events. HMMs are also a succinct representation of speech events; therefore, they require less storage than many other strategies (Lee, 1989). It is most widely applied and said most successful speech modeling technique. Solomon (2001) and Kinfe (2002) also applied HMMs for Amharic language purposes.

Each phoneme model in this work was of 5-state left-to-right HMM. And, the appropriate tools from the HTK toolkit were used for the development purpose. The tools in HTK provide sophisticated facilities for speech analysis, HMM training, testing and evaluation.

1.4.4. Other Knowledge sources

Apart from the acoustic modeling quality, Lee (1989) has mentioned that the performance of a speech recognizer is a function of several variables: the quality of the language modeling, the constraints imposed by the grammar and the inherent confusability of the vocabulary. Thus, a pronunciation dictionary and a language model had to be developed in line with the aforementioned points.

The developed language model was statistical instead of rule based for the limitations under the circumstances and many systems in literature were supported by such language models. In particular, a probabilistic bigram model, a specific form of probabilistic Finite State Grammar language models, was constructed because it is widely used, easy to implement and it is also said to be consistent with the structure of an HMM.

The basic units of recognition used were phonemes. Phonemes were opted because they are highly trainable and sharable. Since there are only 39 phonemes in Amharic, they can be sufficiently trained. In addition, if a user wants to add new words, only the vocabulary and the grammar need be modified, no further training and retraining of the models (Lee, 1989). Thus for large vocabulary continuous speech systems, they were only appropriate units. Finally, kinfe (2002) has also demonstrated that phonemes produce better results for Amharic speech.

1.4.5. Evaluation and Testing Techniques

The most important and the commonest testing parameter used in evaluating speech recognition systems is accuracy of recognition. Continuous speech recognizers may commit three types of errors: substitution, deletion and insertion. Substitution errors result when an incorrect word is recognized in place of the correct one. Deletion error is said to have occurred when a word is omitted from the recognized sentence. An insertion error arises when an extra (untold) word is added in the recognized sentence. Therefore, the performance of the intended recognizer was tested using the test data in light of these three errors at word and sentence levels.

The level of constraint of the language model was evaluated using a common measure called perplexity. It is a measure of the constraint imposed by the grammar, or the level of uncertainty given the grammar. This measure is particularly useful when comparing a system with others.

1.5. Applications and Advantages of Speech Recognizers

Speech recognition technology has got a wide range of applications. Any task that involves interfacing with a computer can potentially use Automatic Speech Recognition. The following areas, however, are summaries of the commonest ones that enjoy the benefits ASR.

- Telephone systems (automatic phone dialing)
- Dictation systems like automatic typewriter that is activated and work by voice (speech)
- As an assistive technology for disabled people like the handicapped that cannot use keyboard or any pointing devices. Even it can be an ideal tool for blind people if it is combined with text to speech systems.

> *Professionals with repetitive stress syndrome (RSS), motor disabilities, and visual impairments view speech recognition as a means of liberating them from dependence on other people. Hands-free dictation allows them to create business and personal documents. Dictation systems are used to a lesser extent by hearing impaired people to follow courtroom proceedings and other primary oral activities (Markowitz, 1996).*

- Computer Aided Instruction systems do make use of ASRs. For developing countries in particular, where majority of the people are illiterate, ASR can help solve many problems like on education, communication, storage and retrieval of orally available knowledge etc.
- Information retrieval is one practical area of utility for speech recognition. For instance, simple inquiries about bank balance, movie schedules, phone call transfers, train and flight schedules can be handled by small-to-medium sized vocabulary, speaker independent speech systems and through phones.
- News agencies can benefit from such systems - the speech of a speaker can be directly transcribed onto paper, while he/she is speaking. This was earlier done by hand after the speech is recorded on tape, which consumes resources (time, money, human and material recourses). The quality of the transcription is also at risk due to the involvement of third person.
- The health care industry is another beneficiary of speech recognition systems.

According to Markowith (1996), "using speech recognition to generate reports … actually takes less of the doctor's time, requires no transcription, and is available immediately. So that if the patients gets wheeled into the operating room the report is there rather than two weeks latter."

- Courts and legal institutions do benefit from the application of ASR system. Direct and proper transcription of the speech from the defendant, the testimonials and later of the judges – on a case, for example – is indispensable and is possible by an ASR.

In general, apart from being a natural way of interfacing with machines (like PCs, Telephones, Television etc) ASR renders the following merits: helps to speed up inputting information (much faster than using the ubiquitous, keyboard even for the fastest possible typists); avoid pains related to typing (like repetitive stress injury); using ones voice, the hands and the eyes are free and movement is unconstrained. The same can't be said of systems that rely on keyboards, mice, or touch screens for input.

According to Markowitz (1996), companies are increasingly choosing to use and develop systems with speech recognition interfaces citing various benefits, including:

- Increasing productivity
- Gaining rapid return on investment
- Access to new markets – can help render 24-hours service
- Environment control – like for disabled people or eyes and hands busy working environment
- Speech is natural to human and would be ideal if we can have it on serving machines.

1.6. Scope and Limitations

This research work was primarily intended to investigate the possibility of large vocabulary, speaker independent and continuous speech recognition for Amharic language. It never tried to develop any specific application.

It was constrained by frequent power interruption. The recognition tool used to take an average of thirty hours and it was a common phenomenon to enjoy any abrupt power failure. Shortage of time was also another constraint. This whole work was supposed to be done in a three and half months. Had there been more time, more performance optimization tasks could have been done. Other related tasks like speaker adaptation could have also been attempted.

1.7. Organization of the thesis

This paper is organized into five chapters. The first chapter gives background information about ASR and its application areas. It also justifies the need for the study, presents the objectives of the study and discusses the methodology followed throughout the work. The second chapter introduces the Amharic phonetics and related features. It also deliberates on the fundamentals of ASR and the various dimensions of ASR. The third chapter deals with the basics of HMMs and provides an overview of HTK and the accompanying used tools. Chapter 4 discusses the design and implementation details of the Amharic Speech Recognition System and the analysis made based on the findings. The last chapter presents the conclusions drawn and the recommendations made.

Chapter 2

Fundamentals of Speech Recognition, Dimensions of ASR and the Amharic Phonemes

2.1. Introduction

It is no dubious that understanding the underlying principles and fundamental concepts of speech recognition is necessary. In addition, knowledge of the basic components that comprises a speech recognition system specifically that of HMM based recognizers is indispensable. Along this line, section 2.2.is committed to lay a theoretical foundation of speech recognition.

It is also true that there are major constraints hindering the development of free and natural human speech recognizers for any language. Section 2.3.is meant to deliberate on such constraints; especially because this study is claiming to investigate the possibility of developing unconstrained speech recognizer for Amharic, by those major challenges.

Finally, it is useful that the important features of the language, as related to speech recognition, should be properly dealt with. Accordingly, the nature of the basic units of Amharic, their contextual behavior and the way they are transcribed are briefly discussed in section 2.4.

2.2. Fundamentals of speech recognition

The underlying assumption behind any recognition system is that the waveform of a speech signal that comes out of a speaker's vocal apparatus is a realization of the concept that was in the form of symbols in his/her mind. When a source conceives an idea to speak out, it was understood symbolically. The moment it gets out to the channel, it materializes in the form of speech signals or sound waves. Thus, one direct and possible approach for a computer based speech recognition system to recognize an utterance is inferring the original symbols from the speech signals (Young et.al, 2002).

To effect this reverse operation of recognizing the underlying symbol sequence given a spoken utterance, first the speech waveform should be digitized and important features must be extracted from the digitized format. Then the extracted features out of the continuous speech waveform should be converted to a sequence of equally spaced discrete parameter vectors (Ibid).

The next requirement for a computer-based speech recognizer is to make estimation of the possible symbol sequences that would result in information carried by the sequence of vectors. Acoustic models can help such speculations about the probability of the observed symbol sequences given the models. Then a language model can be used to formulate and tell the prior probability of estimated word sequences.

The final role of the recognizer is to effect a mapping between sequences of speech vectors and the wanted underlying symbol sequences. Modeling techniques and matching algorithms will be used to find out the required association between hypothesized sequences of words and the uttered speech. This module is referred to as acoustic modeling or statistical pattern recognition or acoustic pattern matching. One important approach of acoustic modeling, employed in this study – Hidden Markov Modeling - is fundamentally probabilistic. It measures the association of utterances to reference patterns by the application of statistical models.

Mathematically speaking:

> *The standard approach to large vocabulary continuous speech recognition is to assume a simple probabilistic model of speech production where by a specified word sequence, W produces an acoustic observation sequence O, with probability P(W/O) (Rabiner and Juang, 1993).*

That is, a model reckoning the probability of hypothesized word sequences in generating the observed sequence from the uttered speech is assumed to serve the purpose.

This means, through speech preprocessing (spectral analysis) a sequence of vectors or observation symbols $O = O_1, O_2,\ldots O_T$, which is a representation of the speech signal, can be obtained. Provided a vocabulary V of all the words that can be uttered, then, mathematically the speech recognition problem gets down to finding the word sequence W having the highest probability of being spoken or that best match the observed sequence (Wiggers, 2001).

Thus, for a given acoustic evidence **O**, what needs to be solved is:

$$\hat{W} = \arg \max_{W} P(W / O) \tag{2.1}$$

Unfortunately, this equation is not directly computable since the number of possible observation sequences is sheer inexhaustible, unless there is some limit on the duration of the utterances and there is limited number of possible observation symbols. But using Baye's rule equation (2.1) can be written as:

$$P\left(W/O\right) = \frac{P\left(O/W\right) * P(W)}{P(O)} \tag{2.2}$$

Since **P(O)** is equal for all, the above equation can be reduced to:

$$P\left(W/O\right) = \arg\max_{W} P\left(O/W\right) * P(W) \tag{2.3}$$

The first term in equation (2.3), **P(O/W)**, is generally called the acoustic model. It estimates the probability of a sequence of acoustic observations, conditioned on the word string (string sequences of models). The second term, **P(W)**, is known as the language model, and it describes the a priori probability associated with a postulated sequence of words (Rabiner and Juang, 1993). Therefore, the goal of any speech recognition system that uses a statistical pattern recognition approach is quantified down to resolving these two models. Consequently, a speech recognizer can be said to possess three components: spectral analysis, language modeling and acoustic modeling and decoding. In the next subsections, these three components will be described.

2.2.1. Spectral analysis: the front-end/Signal Processing

The commonest element of all recognition systems is the front-end signal processing, which converts the speech waveform to some type of parametric representation for further analysis and processing (Rabiner and Juang, 1993).

And the most important parametric representation of speech is the short time spectral envelope[1]. Spectral analysis methods are considered as the core of the signal-processing front-end in a speech-recognition system.

Generally, the purpose of spectral analysis is to derive the feature vectors used to characterize the spectral properties of the speech input. Samples will first be taken from the waveform[2] and will be converted into a sequence of parameter vectors at a certain frame rate[3]. This sequence of parameter vectors approximate the exact representation of the speech waveform (if appropriate sampling frequency and quantization level is taken),

---

[1] A wide range of other possibilities exists for parametrically representing the speech signal like the zero crossing rates, level crossing rates and other related parameters.

[2] The speech waveform is assumed to be in digital form through out this work, the task of which is handled by the inputting device

[3] A frame rate of 10 ms is usually taken, because a speech signal is assumed to be stationary for about such long.

based on the assumption that for the duration covered by a window signal (typically 10 - 50ms), the speech waveform can be regarded as being stationary.

Although this is not strictly true, it is a reasonable approximation. This also helps to reduce the required storage and processing of otherwise prohibitively large amount of speech data (Wiggers, 2001).

A segment of a waveform that is used to determine each parameter vector is referred to as a window. The window size and the frame rate are independent. Normally, the window size will be larger than the frame rate, so that successive windows will overlap to make up for discontinuity in the signal introduced by the sampling of the signal (Ibid).

The two dominant methods of spectral analysis are Linear Predictive Coding (LPC) and Mel Frequency Cepstral Analysis[1].

LPC lies on the assumption that the space between the vocal chords (a buzzer), called the glottis, produces the speech signal (the buzz), which is characterized by its intensity (loudness) and frequency, which determines the pitch of the sound. The vocal tract, that is the combination of the throat and the mouth, forms a tube, which is characterized by its

[1] Mathematical details are omitted here and interested reader may refer Rabiner and Juang (1993) and the references.

resonance, called formants. LPC analyzes the speech signal frames by estimating the formants, removing their effects from the speech signal intensity and frequency of the remaining buzz (Wiggers, 2001).

The process of removing the formants is called inverse filtering, and the remaining signal is called the residue. The basic intention of LPC is to determine the formants from the speech signal. This is done by a difference equation, called a linear predictor, which expresses each sample of the signal as a linear combination of previous samples. The coefficients of the difference equation, the prediction coefficients, characterize the formants (Ibid).

The Mel-frequency cepstral analysis bears the concept of power spectrum. Power spectrum of a speech signal describes the frequency content of the signal over time. The analysis starts with a process called Discrete Fourier Transform (DFT), which computes the frequency information of the equivalent time domain signal (Wiggers, 2001).

However, as a speech signal contains only real point values, a real-point Fast Fourier Transform (FFT) will be performed for increased efficiency. The resulting output contains both the magnitude and phase information of the original time domain signal. Here a different scale called Mel-scale is employed instead of a linear scale (Wiggers, 2001). MFCC features along with their derivatives are now accepted as the standard front-ends in

ASR systems Gu (2001).

2.2.2. Language Modeling

Equation 2.3.depicts that one of the two determining factors in reckoning the maximum a posteriori probability of an observation sequence is the prior probability P(W). The goal of any statistical language model is, therefore, to provide an estimate of these probabilities of word sequences for the given recognition task.

Small vocabulary recognition systems generally do not rely heavily on language models to accomplish their tasks because they are mainly used in command and control applications where the vocabulary words are essentially acoustic control signals that the vocabulary has to respond to plus considering all combination of words is not that difficult.

A large vocabulary speech recognition system, however, is generally critically dependent on linguistic knowledge. Hence, incorporation of knowledge of the language, in the form of a language model, is essential for large vocabulary systems (Rabiner and Juang, 1993).

Such language models may incorporate syntactic or semantic constraints. Often, when only syntactic constraints are used, the language model is called a grammar. Usually, they are represented in a finite state network so as to be integrated into the acoustic model in a straightforward manner (Rabiner and Juang, 1993). This study has employed one such model, thus, a brief explanation of the existing common types of models is warranted and

follows here.

Some language models may be designed in such a way that their finite state grammars produce the same set of sentences with the intended task. They are committed to the sentences in the application or the task. Such grammars will be too simplistic and have low perplexity[1]. However, the long-range goal of many systems is and should be to ensure high perplexity (looser grammar) and that supports large vocabulary (Lee, 1989).

Some other language models may be of simple grammar that specifies only the list of words that can legally follow any given word. Such model is referred to as word pair grammar. In using this grammar, all the HMMs for all the words in the vocabulary will be put in parallel. A null transition will be used from the last state of a word, say X, to the first state of the other word, say Y, if (X, Y) is a legal/possible word pair. The transition probability of this null is 1/N, where N is the number of words in the vocabulary. That is, equal probability is assigned for all transitions of legal pairs (Rabiner and Juang, 1993).

The other variation of grammars is termed as n-gram language model. In this model there is a logical assumption that the probability of a word sequence W can be reasonably computed as:

---

[1] Perplexity refers to the degree of constraints imposed in the language or the grammar.

$$P(W) = P(W_{1,}, W_2 .....W_Q = P(W_1) * P(W_1/W_2) * P(W_3/W_2, W_1) .....P(W_Q/W_{1,} W_{2..} ...W_{Q-1}) ....(2.4.)$$

However, it is essentially impossible to reliably estimate the conditional word probabilities, $\mathbf{P(w_j/w_1 \ldots .w_{j-1})}$ for all words and all sequence lengths in a given language. Thus the concept of n-gram comes in here and states that the term $\mathbf{P(w_j/w_1 \ldots .w_{j-1})}$ can be approximated based only on the preceding N-1 words. i.e.

$$P(W_j/W_{1,} W_2 ....W_{j-1}) \sim P\left(W_j \Big/ W_{j-N} .....W_{j-1}\right) .....forj > N \qquad (2.5)$$

Even n-gram probabilities are difficult to estimate reliably for all but N=2 or possibly N=3 (Rabiner and Juang, 1993).

In most cases, the language model P(w) is estimated from a given (large) text corpus using a simple relative frequency approach,

$$P(W_i/W_{i-1} .....W_{i-N+1}) = \frac{F(W_i, W_{i-1}, ...W_{i-N+1})}{F(W_{i-1}, ...W_{i-N+1})} \qquad (2.6)$$

where F is the number of occurrences of the string in its argument in the given training corpus.

2.2.3. Acoustic Modeling and Decoding

Once the signal has been transformed into a parameterized form it must be recognized, or decoded, and turned into the underlying sequence of symbols. This decoding process requires pattern or models against which unknown utterances can be compared (Odelly, 1995).

The acoustic model, P(O/W), provides the probability that the speech data was observed for a given word sequence. In principle, the required probability distribution could be found by obtaining many examples of each word W and collecting the statistics of the corresponding vector sequences. However, this is impractical for large vocabulary systems and instead, word sequences are decomposed into phonemes (Young, 1996).

It is also difficult to directly estimate the joint conditional probability $P(O_1, O_2 ...., O_n / W_i)$ from examples of spoken words due to the dimensionality of the observation sequence O. Thus a parametric model such as a Markov model can be assumed and the estimation from data will be possible. This is because the problem of estimating the class conditional observation densities $P(O/W_i)$ is replaced by the much simpler problem of estimating the Markov model parameters. Thus in place of W_i, HMMs of the base units like phonemes or words will be used.

Finally, probabilities for word sequences are generated as a product of the acoustic and language model probabilities. The process of combining these two probability scores and sorting through all plausible hypotheses to select the one with the maximum probability, or likelihood score, is called decoding or search (Ganapathiraju, 2002).

In summary, when a test utterance comes in, the signal will be analyzed and encoded as sequence of speech vectors, as discussed in section 2.3.1.3. Then possible matching

sentences will be hypothesized, the corresponding words, phonemes and phoneme models will be formulated from the grammar network. The probability of each sentence or word sequence P(w) is known, as discussed in section 2.3.1.2. Then, the probability of a sequence of words to produce the input sequence P(O/W), the acoustic model, will be calculated. Finally, the a priori probability P(w) will be multiplied with the posteriori probability P(O/W), and the one with the maximum probability will be selected as a recognized sequence.

2.3. Dimensions of Constraints in speech recognition

The target of any speech recognition system development endeavor is to realize unconstrained speech recognizer that works for any speaker, and in any way that the user would like to speak to it. The success of few current systems, however, is never without the imposition of one or more of the following constraints: speaker dependence, isolated word, small vocabulary, or constrained grammar (Lee, 1989).

These constraints need to be alleviated for a system that would claim to get close to the ultimatum of automatic speech recognition. As the intention of this study is to investigate possibilities of reaching at this goal for Amharic language, a bit more detailed discussion of the aforementioned constraints is only justified. Accordingly, the very distinction between the constraints with their counterparts, the problems that warrant them and progresses that have been made towards relaxing them will be reviewed in subsequent sections.

2.3.1. Speaker Independence vs. Speaker Dependence

A speaker dependent system is developed to operate for speaker/speakers that are involved in the training. It uses speech from the target speaker to learn the model parameters of the speakers' voice through training. Such systems are usually easier to develop, cheaper to buy and more accurate, but not as flexible as speaker independent systems (Philip and Young, 1987).

Lee (1989), has pointed out that speaker dependent systems are useful for some applications. However, they have many problems – the training is inconvenient and time taking to the user; a large amount of processing is required; certain applications may involve only multiple speakers; certain applications, like telephone directory assistance, may not tolerate the training delay; considerable additional storage is needed if each speaker's parameters are to be stored separately; and as a speaker's voice may change over time due to stress, fatigue, sickness, or variations in microphone positioning, the system is in trouble.

Speaker independent systems, on the other hand, are capable of recognizing speech from any new speaker. They are designed to be used by any user, with no enrollment or prior training. Such systems in general are said to be most difficult to develop because those applications must recognize large, heterogeneous populations of speakers (Markowitz, 1996). However, they are more flexible and natural.

Despite many efforts, no recognizer has yet achieved a reasonable accuracy on a large speaker-independent task, compared to the speaker independent ones. Researches and experiences also reveal as a rule of thumb that for the same task, speaker dependent systems will have three to five times the error rate of speaker-dependent ones (Lee, 1989).

The difficulty basically arises from the very fact that most acoustic properties and parametric representations of speech are highly speaker independent. Thus a set of reference patterns suitable for one speaker may perform poorly for another speaker. To overcome this problem, it is useful to learn the characteristics of a speaker and perhaps adapt accordingly. It is also desirable to use a large number of speech parameters (features). These require greater amount of training data and a modeling technique that can account for many parameters. As a result efficient and automatic algorithms are demanded (Ibid)[1].

There are three general approaches towards speaker independence (Lee, 1989). The first approach stands on the assumption that it is possible to find invariant parameters from speeches of (expert) readers and if these invariant parameters can be obtained, speaker-independent recognition could be as easy as speaker-dependent recognition. Thus the

[1] In this work, HMMs are used as they are known to be capable of robust modeling of speech. Efficient algorithms do also exist for accurate estimation of HMM parameters.

approach employs knowledge engineering techniques to find perceptually motivated parameters that are relatively invariant between speakers.

The second approach advocates the use of multiple representations for each reference to capture the between-speaker variations. It requires many speakers utter each word in the vocabulary. These multiple examples will be divided in to several clusters, and a prototype will be generated from each cluster.

However, Lee (1989) has indicated that both these two approaches have been experimented and produced good results only for very limited tasks, but has not been successfully extended to large-vocabulary tasks.

The final approach is speaker adaptation. It begins with an existing set of parameters and a small number of adaptation sentences from a new speaker. These sentences are used to modify the parameters so that they are adjusted to the new speaker. However, the adaptation process should be rapid non-intrusive; otherwise it degenerates to speaker dependent systems.

This research work is meant to investigate the possibility of developing speech recognizers that are of speaker independent. The ambition is that any user, regardless of sex, age, or circumstances variations can speak out to the systems and get recognized. The speaker adaptation approach was attempted for the purpose.

2.3.2. Continuous vs. Discrete Speech Recognition

Discrete or isolated speech recognition systems are systems that require the speaker to pause briefly between words. They operate on a single word at a time. The beginning and end points are easier to find and the pronunciation of a word tends not to affect others. However, discrete speech is an unnatural way of speaking that many people find it difficult. It also renders little advantage as to improving the desired level of inputting speed, little fast from typing. In addition, the speaker will be bound to think slower as he/she has to speak so – it breaks the train of thought.

Continuous speech recognition systems operate on speech in which words are not separated by pauses. Speech is said to be continuous when it is uttered as a continuous flow of sounds with no inherent separations between them (Markowitz, 1996). And the recognizer must utilize special methods to determine utterance boundaries.

Continuous speech recognition is more difficult than discrete recognition for three reasons (Lee, 1989): it is difficult to find the start and end points of words (word boundaries), co-articulation effects are much stronger, and content words (like nouns and verbs) are often emphasized while function words (like articles) are poorly articulated. Besides, continuous speech recognition is characterized by massive acoustic variability, intra-speaker and inter-speaker variability. As a result, error rates increase drastically from isolated word to continuous speech.

In spite of the aforementioned challenges, this study attempted to develop and demonstrate continuous speech recognizer due to the enormous benefits that it renders. That is also the dream of any automatic speech recognition system in any language.

2.3.3. Small Vocabulary vs. Large Vocabulary

Vocabularies (or dictionaries) are lists of words or utterances that can be recognized by the speech recognition system. Despite the fascination with unlimited vocabularies there are applications that involve much smaller vocabularies. A phone dialing system for instance, requires fewer than twenty words and most manufacturing applications use fewer than hundred words (Markowith, 1996). On the other hand, dictation systems require large vocabulary.

Generally, smaller vocabularies are easier for a computer to recognize, while large vocabularies are more difficult (Young, 1996). As the vocabulary size gets larger, the number of confusable words grows substantially. Besides, with small vocabularies, each word can be modeled individually, because it is reasonable to expect sufficient training for the handful of words. It is also possible to store the parameters of each word model separately. However, in large vocabularies it is difficult to train each word explicitly because neither the training nor the storage is available[1]. The other problem relates to the complexity of search. For small vocabularies, it is possible to perform optimal search;

[1] Thus sub word units must be identified and used despite their disadvantage – they fail to capture co articulation effects.

however, for large vocabularies, pruning will be necessary, which introduce search errors, affecting accuracy.

According to Lee (1989), large vocabulary typically means a vocabulary of about 1000 words or more. This research envisioned large vocabulary continuous speech recognition systems[1]. And the training data used was made of around 23000 words where as the various test data were comprised of between 2000 – 5000 words.

2.3.4. Presence or Absence of Language Models

Language models usually buttress speech recognition systems that would like to increase their accuracy by anticipating or limiting what can be said at any given time. When speech is produced as sequence of words, language models or artificial grammars can be used to restrict the permissible combination of words. As discussed in subsection 2.2.1.2.they are primarily used to assign probabilities to possible sequence of words. Thus language models ranging from those that are more general (and approximating natural language) to highly constrained ones can be specified. On the other hand, some recognizers may not have any of such knowledge sources.

In general, the performance of any Speech Recognition systems is often challenged by the following sources: different speakers, continuous speech, and the perplexity of the task

[1] Phonemes will be used as basic units of recognition, which are appropriate for large vocabulary speech recognizers than words.

(large vocabulary). This research investigated ways of overcoming all these constraints. Although performance of a small vocabulary, isolated speech or speaker-dependent system has achieved the practical useful level, a general purpose large vocabulary continuous speech and speaker-independent system is still far from requirement for a wide spread applications. This is especially true for Amharic language.

2.4. Amharic Phonetics and the writing system

This section highlights on the Amharic sounds and the writing style in relation to the spirit of this work in particular and speech recognition in general.

2.4.1. The Amharic Phonemes

Amharic is a very typical Ethiopian language, having six of the eight phonological markers and sixteen of the eighteen grammatical features that may characterize the Ethiopian language area (Bender et.al., 1976).

The language is primarily comprised of 38 phonemes – 7 vowels and 31 consonants (Baye, 1997)[1]. One additional consonant /v/ is inherited and included summing up to a total of 39 phonemes. Below is a summarized discussion of these components, with more emphasis to features that affect speech recognition.

---

[1] Appendix 'A' contains a list of all the Amharic phonemes, their phonetic symbols and example words

2.4.1.1. The Amharic consonants

Appendix B, presents all the Amharic consonants based on the three categorizing factors - *point of articulation*, *manner of articulation*, and the presence or absence of *voice* (Clark et al., 1985).

The five glottalic consonants (p', t', s', c', k') are the most striking features of the Amharic phonemes. They are known as 'explosives' or 'ejectives' and correspond to the ordinary consonants p, t, s, c, and k. The difference between the ejectives and the plain consonants is very important. The frequency of occurrence of ejective in running speech is only 5 percent of all the consonants but they make distinct impression (Bender et al., 1979).

The other important feature of the language is the presence of palatal consonants š, č, ž, c', j, and ñ. These correspond to the dental consonants s, t, z, t', d, and n respectively. The total percentage of palatals in running text is about 13 percent of all the consonants (Ibid).

The rounded or labialised consonants produced by rounding the lips while pronouncing them (k^w , k'^w , g^w , m^w, f^w, b^w and h^w) occur often in Amharic, but are rare in other semitic languages.

Another important feature of Amharic phonology is the lengthening in time of the pronunciation of consonants called 'gemination'. The local grammarians call this 'tightening' of consonants, showing that it is thought of as a form of emphasis or stress.

Unlike other languages like English, the rhythm of Amharic is marked mainly by longer and shorter syllables depending on the gemination of consonants, and by certain features of phrasing (Bender et al., 1979). All the consonants of the language except /h/(ሀ) can be geminated (Baye, 1986).

2.4.1.2. The Amharic Vowels

In Amharic, there are a total of 7 vowels - five of the commonest vowels a, e, i, o, and u plus two additional central vowels E and I. The vowel /E/ is by far the most frequently occurring vowel in the language, accounting for about one-third of all the vowel occurrences. The vowels /u/ and /o/ are rounded while the rest are unrounded (Bender, 1979).

Vowels are most usefully described in terms of the *position of the tongue* as they are articulated. The following table suffices to illustrate the position of the Amharic vowel phonemes in the usual articulatory configuration.

	Front	Center	Back
High	I	I	u
Mid	E	E	o
Low		a	

Table 2.1. The Amharic vowels

2.4.1.3. Effect of context on Amharic phonemes

Like many other languages, contextual variation has got its own impact on the very nature of sounds in this language too. Baye (1986) has indicated that Amharic phonemes may be affected by the behavior of the neighboring phonemes, apart from their own place and manner of articulation. Here, mention is made of some selected situations that may be of concern to this study.

A phoneme may totally resemble its neighbor when they occur together i.e. a sound may take after its neighboring sound in a context. Total resemblance occurs when the place and manner of articulation of a phoneme changes; and start being created at the same place and in a similar manner to the influencing neighboring sound.

Eg. /Anbessa/ (†¿l]) → [Ambessa] (†Nl])

/Anbet'a/ (†¿l») → [Ambet'a] (†Nl»)

/Kenfer/ (Š¿ÔY) → [Kemfer] (ŠNÔY)

In the examples the sound /n/(ጎ) is turned to /m/ due to its precedence to /b/(በ) and /f/(ፈ). In other cases, any consonant coming before a high vowel like /i/ and /e/, will be turned to high sound.

Eg. /Kiss/ (Šé^) → [Kyiss] (Šé¨^)

/Set/ (\îr) → [Seyt] (\îî¨r)

Sometimes, consonants may get rounded due to the presence of the vowels /o/ (ኦ).

Eg. /T'or/ (½Y) → /T'wor/ (½'åY)

/ZorE/ (□T) → /ZworE/ (□'åT)

In some other situation, phonemes may exchange their positions due to the neighborhood effect.

Eg. /kIbrit/ (ŒnVr) → /kIrbit/ (ŒYlér)

/RIhab/ (Yˆ An) → /IRhab/ (ˆYAn)

It is true that in many other languages like English, one of the pressing problems relates to the phonemes in function words. Such phonemes are usually distorted. Especially in continuous speech the problem is grave. However in Amharic phonemes in function words doesn't appear to be highly distorted. At least in this work, all such words are attached to a stem word that they are made to be separately non-existent. This will partially reduce the problem of developing continuous speech.

2.4.2. The Amharic Writing System

The present writing system of Amharic is taken from Giiz, which was the language of literature in Ethiopia until recent times. Through adaptation and use over a long period of time, the script has undergone changes in shape and number, some being dropped and others added (Bender, 1976).

The writing system consists of a core of thirty-three characters (fidEl) each of which occurs in a basic form and in six other forms known as orders. The seven orders represent syllable combinations consisting of a consonant and following vowel. All in all, there are 231 different characters. It is also true that no capital-lower case distinction is there in the system and the use of different type styles (corresponding to bold face, italics, etc) is not well developed in the system (Bender et al., 1976).

While the sound and the spelling systems of many languages do not correspond exactly, the degree of misfit between sound and spelling in Amharic is not significant. Provided that one knows the alphabets of the Amharic language, the chances of pronouncing a word upon seeing it written for the first time are pretty good. Furthermore, upon hearing a word in Amharic, one has a good chance of spelling it correctly (Clark *et al.*,1985).

In relation to the writing system, Bender(1976) has mentioned three problems:

- Presence of several unneeded symbols (fidEl) - like for the sound /ha/ there are around 5 symbols ሀ, ሐ, ኀ, ኃ, ሃ.
- Many of the syllabic characters are irregular and no standard system of handwriting is there.
- There is no marking for the germination of consonants and there is always ambiguity between six-order symbols with and with out vowel

Chapter 3

The HMMs and HTK

This chapter deals with the well-known modeling technique (Hidden Markov Models) which is employed in this study; and summarizes the HMM toolkit HTK that helps to develop and implement HMM based speech recognizers. The discussion of the toolkit is mainly based on the handbook prepared by Young et.al., 2002 and mention will be made if otherwise.

3.1. Hidden Markov Models

This section presents the basics of HMMs. It begins by explaining what an HMM is and how it operates. Then it explains how they may be used for speech recognition in light of the assumptions taken, the topologies that may be used, the various types of HMMs, the implementation of all the knowledge sources for a speech recognizer, and the three basic HMM problems that should be addressed in speech recognition.

3.1.1. Basics of HMMs

It has been shown that the a priori probability of a sequence of words P(W) and the a posteriori probability of the sequence given an observation P(W/O) - the two models in equation 2.3 - are important and are modeled separately. The former is determined by a language model where as the later is given by an acoustic model.

In most state-of-the-art recognition systems, the hidden Markov model (HMM) is used in the acoustic modeling. HMM provides a simple means to estimate the above conditional probabilities on the basis of finite-state Markov chains (Gu, 2001). It is assumed that a Markov model generates the sequence of observed speech parameter vectors corresponding to each word.

An HMM contains a Markov distribution for transitions across different states, and includes a probability density function at each state that models the probability of the output symbols possible at that state. It is a doubly stochastic state machine that can be fully described by the triple {S, A, B}. Here, S is the initial state probability, $A = \{a_{ij}\}$ is the state-transition probability set, and $B = b_j(O_t)$ is the emission probability distribution (Hamaker, 2002).

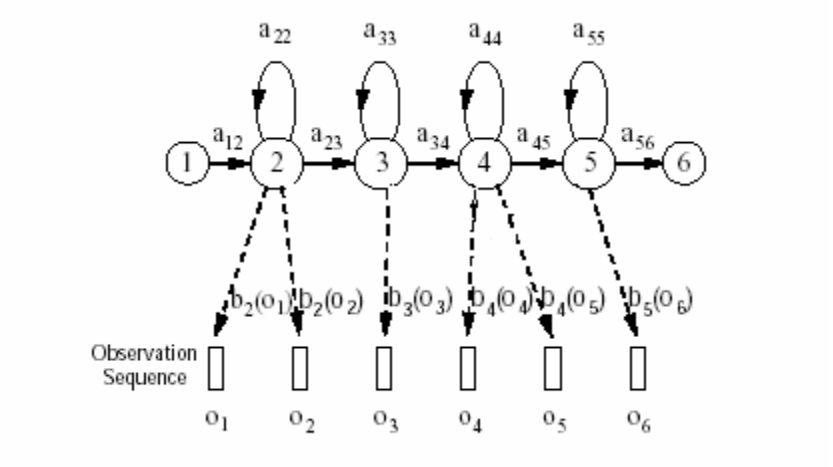

Fig.2.1. A Hidden Markov Model

The model changes state once every time unit. Each time t that a state j is entered, a speech

feature vector O_t is generated from the probability density $b_j(O_t)$. Furthermore, the transition from state i to state j is also probabilistic and is governed by the discrete probability a_{ij}.

3.1.2. HMMs for Speech Recognition

The popularity of HMMs as a model for speech phenomena is owed to the HMMs ability to simultaneously model the temporal progression of speech (speech is usually seen as a "left-to-right" process), and the acoustic variability of the speech observations. The temporal variation is modeled via an underlying Markov process, which is usually hidden; while the emission distribution models the acoustic variability. This acoustic variability may come as a result of differing speakers, channel conditions, stress levels, dialect, accent etc. in the speech training corpus (Ibid). Thus the choice of output probability function is crucial, as it must model all of this intrinsic spectral variability of speech signal (Srivastava, 2002).

In addition, the strength of hidden Markov models in speech recognition is augmented by the existence of computationally efficient algorithms for estimating the model parameters given example observation sequences of known class, called training, and for choosing which model best matches an observation sequence of unknown class, called recognition or decoding (Odelly, 2002).

Given these undisputed relevance of HMMs for speech modeling, it is reasonable devoting the following subsections in discussing how it applies in the area.

3.1.2.1. HMM Assumptions

There are a set of assumptions implicit in the structure of hidden Markov models about the structure of the process that they represent, though all the assumptions are not necessarily be correct(Odelly, 1995).

The first assumption states that the observation accurately represents the speech signal. Normally the observation takes the form of some type of short term (like 10ms) spectra. These will not exactly represent the underlying speech. However it can be said that the speech is nearly stationary over such short periods and the observation is a reasonably accurate representation of the speech. This assumption is important in that prohibitively large amount of data would have to be processed, otherwise (Markowith, 1996).

The second assumption dictates that the likelihood of generating each observation is dependent only upon the state and is independent of all the other observations. This is not typically true of speech since the spectrum tends to change only slowly compared to the frame rate to ensure that the parameterized observations are an accurate representation of the original signal. However augmenting the observations with derivative parameters can reduce the effect of the resulting high degree of correlation between subsequent

observations. When these are used the correlation does not seem to have an adverse effect on the recognition and attempts to model the correlation explicitly have not succeeded yet.

The third assumption underscores that the state transition probabilities remain unchanged.

3.1.2.2. HMM topologies for speech recognition

Most topologies used in speech recognition are based on the assumption that there are three phases in the pronunciation of a phone. In the first phase the vocal tract is changing shape to pronounce the phone, this is called the on-glide of the phone. In this phase there may be some overlap with the preceding phone.

In the second phase the sound of the phone is assumed to be pure and in the third phase the sound is released and the vocal tract starts to transit to the next phone. This is called the off-glide, some overlap with the next phone may occur here. This suggests that at least three states should be used in a phonem based HMM (Wiggers, 2001). In the three states, the first could represent the transition in to the phoneme, the second bears the steady state portion and the third depicts the transition out of the phoneme (Lee, 1989).

Adding more states means introducing more parameters and thus more degrees of freedom. Variations in a phoneme can be modeled more accurate but this also introduces a need for more training data to avoid under training. But a model should not be too large, a five state model does not work for phones that only occupy three time frames, so in larger models there should always be a ‘short-cut’ that can handle the shortest example in the training data (Wiggers, 2001).

3.1.2.3. Variants of HMMs

One way to classify types of HMMs is by the structure of the transition matrix of the Markov chain. Accordingly, there can be many types of HMMS as there are many variations and combinations of the matrix. But two of the common models in speech recognition and related tasks are the left-to-right (directional) and the fully-connected (ergodic) models.

The left-to-right architecture is the most common in speech recognition (Markowitz, 1996). This is mainly due to the presence of strong temporal constraints in speech (Lee, 1989). That is, the property of the speech signals change overtime in successive manner making the left-to-right model more appropriate. In such HMMs, the state transition coefficients have the property that no transitions are allowed to states whose indices are lower than that of the current state i.e. $\mathbf{a_{ij}=0}$, where $\mathbf{j < i}$. As time increases, the state sequence increases or stays the same (Rabiner and Juang, 1993).

Furthermore as a word or even a sentence can be seen as a large string of phones, a left-to-right model structure seems to be necessary, back-loops do not fit very well in this picture (Wiggers, 2001). In the ergodic structure, on the other hand, every state of the model could be reached from every other state and they are non-dirctional (Markowitz, 1996). Otherwise HTK allows the building of HMMs with any desired topology (Young et.al, 2002).

3.1.2.4. Continuous vs.Discrete Observation Densities

HMMs may also be catagorized based on the type of output probability distribution bj(Ot). Accordingly, there can be discrete probability distribution HMMs, continuous density HMMs and tied-mixture HMMs.

Among these variants continuous-density HMM and the tied-mixture HMM, are two of the most popular types of HMM in current ASR systems (Gu, 2001). Such HMMs most commonly use a multivariate Gausian output distribution[1] (Ganapathiraju, 2002). HTK is also defaulted to continuous density systems in preference to discrete systems because, for the purpose of research, they have a number of mathematically desirable properties and, for the purpose of practical application; they are believed to be more robust (Young, 1994).

In discrete HMMs, the observations are characterized as discrete symbols chosen from a finite alphabet, and therefore a discrete probability density with in each state of the model can be used. The problems with this approach are:

- each observation is associated with one output only

[1] Other distributions, such as Laplacians have been used, but have had limited success.

- it doesn't reflect correlation between two discrete unit when spoken in continuous speech.

To use a continuous observation density, some restrictions must be placed on the form of the model probability density function (pdf) to ensure that the parameters of the pdf can be re-estimated in consistent way.

The most general representation of the pdf, for which a re-estimation procedure has been formulated, is a finite mixture of the form (o, μ_k, Σ) i.e.

$$b_j(o) = \sum_{k=1}^{M} c_{jk} N(o, \mu_{jk}), 1 \le j \le N$$

Where o is the observation vector being modeled, c_{jk} is the mixture coefficient for the k^{th} mixture in state j and N is any density (like Gaussian).

3.1.2.5. HMM representation of all the knowledge sources

A central philosophy of HMM based speech recognition is that any knowledge source that can be represented as a hidden Markov model should be represented as one. By representing all knowledge sources as HMMs, the recognition process merely consists of a search in an enormous HMM (Lee, 1989).

If phoneme models and a grammar are used, word and sentence knowledge can be incorporated in to the recognizer.

Each word could be represented as a network of phonemes which encodes every way the word could be pronounced. Then the grammar will be

represented as a network whose transitions are words, and the network would encode all legal sentences. Thus by taking this one large HMM, we can instantiate each word with the network of phonemes and then instantiate each instance of the phoneme with its HMM. It is there fore possible to perform a global search that takes all knowledge into account at every step (Lee, 1989).

In other words a language model can be seen as a Hidden Markov Model by taking advantage of the fact that embedding HMMs into an HMM leads to a new HMM[1]. Thus the word states can be replaced by the corresponding word or phone level HMMs resulting in one huge HMM. In case of phone level HMMs, the phone models have to be concatenated to form a word, before substituting. Now the Viterbi algorithm can be used to find the most likely path through this composite HMM. This path will then lead through a sequence of words that specify the recognized word string (Wiggers, 2001).

The same integrated approach during training does holds. Actually, one major advantage of HMM based modeling over the others is that it does not face word boundary identification problem, during training of the models in continuous speech recognition. Since the state sequence is hidden in HMM, it does not matter where the word boundaries are (Lee, 1989).

---

[1] The first and last states of each HMM are non-emitting entry and exit states. These states provide the glue needed to join models together.

To train the parameters of HMMs, all we need is the word sequence in each sentence. Each word is instantiated with its model (which may be of concatenation of sub word models). Next the words in the sentences are concatenated with optional silence models between words. This large concatenated sentence HMM is then trained on the entire sentence. Since the entire sentence in HMM is trained on the entire sentence, all word boundaries are considered at every frame (Ibid).

3.1.2.5. The three HMM problems

For an HMM to be useful in building speech recognizers, three fundamental problems must be solved (Rabiner and Juang, 1993):

Problem 1: Given a model and a sequence of observations, how do we compute the probability that the model produced the observed sequence?

This problem is a problem of evaluating how well a given model matches a given observation (Ibid). To solve this problem the forward-backward algorithm is used.

Problem 2: Given the observation sequence $O = o_1, o_2, o_3, \ldots o_T$ and the model M, how do we choose a corresponding state sequence $Q = q_1, q_2, \ldots q_T$ which is optimal in some meaningful sense?

Viterbi algorithm is used to solve this problem.

Problem 3: How do we adjust the model parameters so as to account for the observed signal? The Baum-Welch re-estimation algorithm can solve this problem.

3.2. The HTK

HTK is a portable and integrated suit of software toolkit for building and manipulating systems that use continuous density Hidden Markov models. It was developed by the Speech Group at Cambridge University Engineering Department with the intention of having a general purpose platform for research, benchmark testing and product development (Young, 1994).

As HMMs can be used to model any time series, the core of HTK is also similarly general purpose. However, HTK is primarily designed for building HMM based speech processing tools, in particular speech recognizers. It can be used to perform a wide range of tasks in this domain including isolated or connected speech recognition using models based on whole word or sub-word units, but it is especially suitable for performing large vocabulary continuous speech recognition.

The HTK toolkit consists of a set of 10 plus library modules and a set of more than 25 tools (programs)(Young, 1994). Much of the functionality of the toolkit is built into library modules, they ensure that every tool interfaces the outside world in exactly the same way and they provide a programming environment for the creation of custom tools or the integration of recognizer functionality in an application.

A table containing the major library modules with their accompanying functions is attached at Appendix C.

HTK tools are designed to run with a traditional command-line style interface. The layout of the commands is the same for all tools. Each tool has a number of required arguments plus optional arguments. The latter are always prefixed by a minus sign.

In addition to command line arguments, the operation of a tool can be controlled by parameters stored in a configuration file. The main use of configuration files is to control the detailed behavior of the library modules on which all HTK tools depend.

Most tools, especially editing tools, like *HHed* which edits Hidden Markov models, or *HLEd* which edits transcription files also need a script that tells the tool which steps to perform and how to edit the data provided to it. The available scripting commands are tool specific.

The HTK tools can best be described in line with the steps involved to develop sub word based continuous speech recognizers. Thus there are data preparation tools, training tools, testing tools, and analysis tools.

3.2.1. Data Preparation Tools

HSLab is an interactive label editor for manipulating speech label files. It can be used both to *record* the speech and to *manually annotate* it with any required transcriptions. In using this tool one can load a sampled waveform file, determine the boundaries of the speech units of interest and assign labels to them. Alternatively, an existing label file can be loaded and edited by changing current label boundaries, deleting and creating new labels. *HSLab* is the only tool in the HTK package, which provides a graphical user interface.

Although all HTK tools can parameterize waveforms on-the-fly, in practice it is usually better to parameterize the data just once. Thus the tool *Hcopy* is used for this purpose of copying one or more source files to an output file. By setting the appropriate configuration variables, all input files can be converted to parametric form, as they are read-in. Therefore, simply copying each file in this manner performs the required encoding.

HList can be used to check the contents of any speech file and since it can also convert input on-the-fly, it can be used to check the results of any conversions before processing large quantities of data.

HLEd is a script-driven label editor which is designed to make transformations to label files, like translating word level label files to phone level label files, merging labels or

creating triphone labels. HLEd can also output files to a single Master Label File MLF, which is usually more convenient for subsequent processing.

HLStats *can* gather and display statistics on label files and where required and ***Hquant*** can be used to build a VQ codebook in preparation for building discrete probability HMM system.

Using these tools in the end the transcription and the parameterized data files will be ready for training.

3.2.2. Training Tools

HMM definitions can be stored externally as simple text files and hence it is possible to edit them with any convenient text editor. With the exception of the transition probabilities, all of the HMM parameters given in the prototype definition are ignored.

The purpose of the prototype definition is only to specify the overall characteristics and topology of the HMM. The training tools will compute the actual parameters later. Sensible values for the transition probabilities must be given but the training process is very insensitive to these.

An acceptable and simple strategy for choosing these probabilities is to make all of the transitions out of any state equally likely. If segmented transcriptions are available the tools *HInit* and *HRest* provide isolated word style training using the fully labeled data as bootstrap data.

Hinit can be used to provide initial estimates of whole word models in which case the observation sequences are realizations of the corresponding vocabulary word. Alternatively, *Hinit* can be used to generate initial estimates of *seed* HMMs for sub-unit based speech recognition. In this latter case, the observation sequences will consist of segments of continuously spoken training material. *HInit* will cut these out of the training data automatically by simply giving it a segment label.

In both of the above applications, *HInit* normally takes as input a prototype HMM definition, which defines the required HMM topology i.e. it has the form of the required HMM except that means, variances and mixture weights are ignored. The transition matrix of the prototype specifies both the allowed transitions and their initial probabilities. Transitions, which are assigned zero probability, will remain zero and hence denote non-allowed transitions. *Hinit* estimates transition probabilities by counting the number of times each state is visited during the alignment process.

Hrest performs basic Baum-Welch re-estimation of the parameters of a single HMM using a set of observation sequences. HRest can be used for normal isolated word training or it can be used for isolated model training for sub-unit based speech recognition, like HInt.

HERest is used to perform a single re-estimation of the parameters of a set of HMMs using an *embedded training* version of the Baum-Welch algorithm. Training data consists of one or more utterances each of which has a transcription in the form of a standard label file (segment boundaries are ignored). For each training utterance, a composite model is

effectively synthesized by concatenating the phoneme models given by the transcription. Each phone model has the same set of accumulators allocated to it as are used in *HRest* but in *HERest* they are updated simultaneously by performing a standard Baum-Welch pass over each training utterance using the composite model.

HHEd is a HMM definition editor which will clone models into context- dependent sets, apply a variety of parameter tyings and increment the number of mixture components in specified distributions.

To improve performance for specific speakers the tools ***HEAdapt*** and ***HVite*** can be used to adapt HMMs to better model the characteristics of particular speakers using a small amount of training or adaptation data.

3.2.3. Recognition or Testing Tools

HTK provides a single recognition tool called *HVit*e, which uses a token passing algorithm. *Hvite* takes as input a network describing the allowable word sequences, a dictionary defining how each word is pronounced and a set of HMMs. It operates by converting the word network to a phone network and then attaching the appropriate HMM definition to each phone instance. Recognition can then be performed on either a list of stored speech files or on direct audio input.

HVite can support cross-word triphones and it can run with multiple tokens to generate

lattices containing multiple hypotheses. It can also be configured to rescore lattices and perform forced alignments.

The word networks needed to drive *HVite* are stored using the HTK *standard lattice format(SLF)*. This is a text-based format and hence word networks can be created directly using a text-editor. However, this is rather tedious and hence HTK provides two tools to assist in creating word networks: Hbuild and Higher level language (EBNF).

Hbuild allows sub-networks to be created and used within higher-level networks. Hence, although the same low-level notation is used, much duplication is avoided. Besides, *HBuild* can be used to generate word loops and it can also read in a backed-off bigram language model and modify the word loop transitions to incorporate the bigram probabilities.

As an alternative to specifying a word network directly, a higher level grammar notation can be used. This notation is based on the Extended Backus Naur Form (EBNF) used in compiler specification The tool ***HParse*** is supplied to convert this notation into the equivalent word network.

Whichever method is chosen to generate a word network, it is useful to be able to see examples of the *language* that it defines. The tool ***HSGen*** is provided to do this. It takes as input a network and then randomly traverses the network outputting word strings. These strings can then be inspected to ensure that they correspond to what is required. *HSGen* can also compute the empirical perplexity of the task.

Finally, the construction of large dictionaries can involve merging several sources and performing a variety of transformations on each source. The dictionary management tool ***HDMan*** is supplied to assist with this process.

3.2.4. Analysis Tools

Hresult is one HTK tool that compares recognition results with original transcriptions. It uses dynamic programming to align the two transcriptions and then counts substitution, deletion and insertion errors. *HResults* can also provide speaker-by-speaker breakdowns, confusion matrices and time-aligned transcriptions.

Chapter 4

Experimentation

This chapter describes the design and construction of the speech recognizer for Amharic language that is capable of recognizing Amharic speech (sounds). The intention was developing a system that is as general as possible, so that it can be used, with slight modifications and some adaptive training, as a speech interface for many other different applications.

To meet these requirements the recognizer was designed to recognize large vocabulary, continuous speech and is speaker independent. This was implemented using phonemes as base unit. Thus, the system's vocabulary was not limited in any way to some fixed set of words. If a new vocabulary is required to be incorporated, what needs to be done is only modifying the language model and adding the new word to the pronunciation dictionary. This means that the system can easily be extended and adapted to many applications.

The development process was performed on the Microsoft Windows platform using the tools in HTK. Various preprocessing programs and scripts were also used. The discussion of the experiment is presented in accordance to the steps that should be followed while building such speech recognizer. Accordingly, the chapter is organized in to four parts – *preprocessing, Training, Optimization (Enhancement)* and *Evaluation.*

The preprocessing part presents the way the data preparation task was performed, the developed language model and the prepared dictionary. The training portion deliberates on how the acoustic models were developed, the steps followed in training and building the monophphone level recognizer, and the tools with their accompanying scripts used in the process. It also discusses the mechanisms followed in refining and optimizing the monophone recognizer. The final part is on the testing and performance evaluation of the systems at various stages.

4.1. Data Preparation

It is plain enough that the required speech data should be prepared and made available before training the models and building the recognizer. Further, the data should pass through different preprocessing tasks in accordance to the requirements of the various algorithms (tools) in HTK. Thus, the utilized data, the output of the data preparation process, and the preprocessing steps involved are described.

The expected major deliverables of this task were processed data, the transcription (label) files, pronunciation dictionary and bigram language model.

4.1.1. The data base

It is an already established fact that in order to build a robust large vocabulary speech recognizer, all the acoustic (sub-word) models should receive enough training examples. In due course, the many variations that can occur in speech, due to, for example, variation in speed of speech, type of speaker, gender and different accents should be taken into account. It follows that a training corpus should be sufficiently large, consisting of speech samples from wide range of age group. The construction of such huge database is a major undertaking by itself and involves a number of subtasks. It also demands sufficiently large time and big financial support.

Thus, as mentioned in the methodology section, the data base used for this work was a corpus developed by Solomon Tefera. This data base fulfills all the aforementioned requirements and is comparable to the big commercially available corpuses for other fortunate languages (like English). The corpus contained 8000 sentences spoken by 80 people, each having read 100 utterances. The ratio between male and female speakers is almost fifty-fifty. The utterances are also comprised of all Amharic phonemes in many phonetic contexts. In addition to the training data, the corpus contained evaluation, adaptation and development test data.

The database, however, is prepared from utterances made by people whose first language is Amharic. So the recognizer developed out of it might not be well serving for all human

kind in the world except that their mother tongue is Amharic.

Three different data sets were taken from the database - a training set, a development set, which was used for testing and fine tuning during development of the system and an evaluation test set to evaluate the final performance of the system. All the training data were used because such huge data was required to develop speaker independent models Where as one part of the development test data and evaluation data were used. The following table summarizes the type and size of data set used for this work

Data Set	No.(speakers)	No.(sentences)
Training	80	8000
Development Test	15	270
Evaluation Test	15	270

Table 4.1. The data set used

4.1.2. The Phoneme Set

As was already mentioned, the basic speech units in the system were phonemes had to be chosen. However, according to Wiggers (2001), there could be found no such thing as a phoneme set for a language, unlike the situation for example with the characters in the alphabet. Existing sets differ in the number of details they provide. But for Amharic, Baye (1994) identified 39 phonemes for Amharic that are used in standard dictionaries. All these sounds exist in the database including a rounding sound ‘uA’. Thus, all of them were considered in the selected phoneme set for this work.

However, the transcription of the phonemes in the corpus was primarily in a font known as Ethiop. And the phonemic representations of the font were not suitable for the tools in HTK toolkit. Thus they were renamed and modified for convenience in later activities and as per the requirements of HTK[1].

Furthermore two other models were added. The first, sil, models longer periods of silence. These silences occur between sentences or when a person is not speaking at all. The second phoneme, sp, also represents silence, but only periods of short duration. This is the kind of silence that occurs between words. The former represents periods of pure silence that can greatly vary in length. It usually demarcates sentence boundaries. While the later represents optional periods of silence shorter than the duration of a phoneme, and are usually found between words.

Appendix A. depicts the phonemes used in their original notation, the used notation in this experiment, an exemplary word for each phone and phonetic transcription of the words.

4.1.3. Dictionary preparation

A program was required and written in order to create the word list out of the text and to prepare the pronunciation dictionary. No external and standardized pronunciation

[1] The full set of the converted and used phonemes in this report are presented at appendix A.

dictionary was used to develop the required dictionary. The program creates a list of all the words in the database. Using the prepared sorted word list, it produced a pronunciation dictionary, containing the words and their associated phone level pronunciation in parallel.

Then, using the tool HDman in HTK and the produced dictionary, two different new dictionaries were created – one with a short pause at the end of every pronunciation and the other without it.

HDman takes a name of the new dictionary to be created and the original dictionary as its parameters plus other optional parameters. The following command was executed to generate the dictionaries, in batch:

hdman -w wordsort.txt -n monophones1.txt -l dlog1 diction1.txt diction.txt

hdman –g d\global.ded -w wordsort.txt -n monophones2.txt -l dlog2 diction2.txt diction.txt

Monophones1.txt and monophones2.txt, which are preceded by the –n format specifier, contained lists of all the unique monophone sounds in the database and are generated by the tool. One of the phonemes lists generated, monophone2, contained '**sp'** as a phone (a model) and the other, monophone1, did not. At the end of both the phoneme lists, '**sil'** is added manually to model the longer silence.

The global.ded script was used to attach sp at the end of each entry of the dictionary and was of the follwing form:

AS sp
MP sil sil sp

The AS command append the sp model at the end of each pronunciation. The model was automatically appended at the end of every word out of a reasonable assumption that there would usually be a very brief pause between words. If the dictionary contained any silence markers then the MP command will merge the sil and sp phones into a single sil.

4.1.4. Transcription Files

Many of the operations performed by HTK assume that the speech is divided into segments and each segment should have a name or a label. The set of labels associated with a speech file constitute a transcription. Two types of transcription files were expected to be prepared out of the utterances: word level and phone level transcriptions. This was because, as the system is phoneme based, later during training examples of the phones were needed to adjust the parameters of these models. That is, to train a set of HMMs every file of training data must have an associated phone level transcription.

However, the examples in the training data set consisted complete sentences, which were strings of phones. Thus to ensure that the right part of an utterance is used to train a model, a transcription of the utterances at phone level was needed.

Besides, to evaluate the performance of the final system, textual transcriptions of what was really said in the training, test and evaluation utterances were needed.

These phone level transcriptions were created from the word level transcriptions. A program was written to prepare the word level transcription first. It read each word from the transcription text, put them in a separate line and changes the appropriate fonts as per the selected phone notation.

Then a file called Master Label File (MLF)[1] containing all the transcriptions of the words in the utterances, indexed by the file name of the utterances was produced. All the utterances had unique file names and the file was prepared in a label file format required by HTK.

Then using the already created pronunciation dictionary and the above phone level MLF, the phone level transcription was created by the tool HLed in HTK. It takes the word level MLF, the pronunciation dictionary and an editing script as its arguments.

---

[1] MLFs are index files holding pointers to the actual lable files. Thus they allow large sets of files to be stored in a single file and a single transcription file to be shared by many lable files (Young, 2002)

The following batch file was run containing two HLed commands:

*Hled –l * -d diction1.txt –i phones1.mlf mkphones1.led wordmlf.txt*
*Hled –l * -d diction2.txt –i phones2.mlf mkphones2.led wordmlf.txt*

As a result two types of phone level transcriptions were prepared, one with sp at the end of each word and the other with out. The first phone level transcription, phones1, was produced out of the dictionary with out sp, but diction2 had sp at the end of each word and same with the resulting phone level transcription, phones2.

4.1.5. Language Model

Language model is a crucial part of a speech recognition system. It tells the system how likely it is that a certain string of words is uttered. By so doing, it imposes a grammar onto the system. Furthermore, as discussed in section 3.1.2.4., the language model is the glue that holds together the set of acoustic models in the word network that is ultimately used for recognition.

The Language Model used in this work was a backed-off bigram language model. It was developed using two of the HTK tools HLstats and HBuild. HLstats was primarily used to generate the bigram probability matrix. It reads in the sorted word list and the word level MLF.

Using the word level transcriptions and statistics on the number of occurrences of each word and each combination of two words, the probability matrix was prepared[1].

These statistics were then used to create ***backed-off bigram language*** models for the training, test and evaluation sets, using the HBuild tool which translated the gathered statistics into HTK Standard Lattice Format[2], that are used for storing word models and multiple hypotheses from the output of a speech recognizer. A sample of the generated bigram in lattice format is attached at appendix J.

HBuild and HLstats were invoked by the following commands:

hlstats -o -b bigram wordsort.txt wordmlf.txt

hbuild -n bigram wordsort.txt outlat

The perplexity of the developed language model was then measured using HSgen algorithm.

HSgen -q -s -n X outlat dicition3.txt

The –n option used to state the number of sentences, X, to be generated out of the model. The default is 100 sentences. But sentences were generated from 100 to 1000 incrementing

---

[1] For the mathematical details on how the statistics is generated, one may consult the HTK handbook.

[2] Standard Latice Format (SLF) is a low level notation of word networks in HTK, in which each word instance and each word to word transition is listed explicitly.

by 100 and an average perplexity of 52.5 was obtained with average entropy of 5.67[1]. This figure is good and is in line with expectation. Literature dictate an average perplexity of 60 or below that.

4.1.6. Coding and feature vector extraction

The other important task in data preparation is the extraction of feature vectors i.e. parameterizing the raw speech waveforms into sequences of feature vectors. This can be done on the fly during training, as is the case during live recognition. But for training purposes it is reasonable to do this once and for all and save the feature vectors in a file. Because each utterance is processed a number of times during training and this helps reduce later processing costs.

HCopy is a tool for translating audio files to feature vector files using one of the many variants of either linear predictive coding or Mel scale cepstral coefficients. It requires as its parameter, one script file[2] listing the file names it has to copy and the new file names they should be copied to in parallel as its parameters.

---

[1] Entropy is the amount of information required to go to the next unit in the language model

[2] Script files refer to files containing a list of other files and has got the extension of .scp – this is also how it is understood in the HTK toolkit.

The script file was of the following format:

```
Tr01001.wav     Tr01001.mfc
Tr01002.wav     Tr01002.mfc
Tr01003.wav     Tr01003.mfc
(etc.)
```

The other input of the tool was a configuration file that tells all the conversion parameters. The utterances were encoded to Mel-frequency cepstral coeffient vectors. This choice was based on literature and earlier experiments. Each vector contained twelve cepstral coefficients with log energy and delta and acceleration coefficients added resulting in 39 dimensional feature vectors. A sampling rate of 10 ms was used with an overlapping window of 25 ms. The configuration parameter used is attached at appendix H.

The script file containing list of the training files was stored in train.scp and the above configuration file was stored in config1.cfg. Then the following command was executed to parameterize all the 8000 training speech utterances.

```
HCopy -T 1 -C config8 -S train.scp
```

The test data was also coded in a similar fashion that by then all the data were ready to train the models.

4.2. Training the Models

In the previous activities all the required resources for training the recognizer and developing the models were prepared. Here the tasks performed in developing the recognizer and the techniques followed in refining and improving performances is explained.

4.2.1. HMM Prototype

The very first step in training HMMs is the selection of a Hidden Markov Model topology for the acoustic models. This was done by defining a prototype model. Since a phoneme based recognizer was built a model represented a phoneme. The parameters of the model are not important because they are to be modified later during training. However it helps to define the models. Thus the means and variances of all the states in the model were simply assigned a value of 0 and 1 respectively.

The topology of the model consisted start and end states and three emitting states, using single Gaussian density functions. The states were connected in a left-to-right way, with no skip transitions.

A five state topology was chosen because it resulted in better performance for other language's sounds that are even longer than the Amharic phonemes. Single mixture was used first and later the mixture was incremented. The prototype model used was adopted from the HTK handbook and it is attached as it is in appendix D.

Each vector was of length 39 which was computed from the length of the parameterized static vector (MFCC_0_D_A) plus the delta coefficients (+13) and the acceleration coefficients (+13).

4.2.2. Initial models

Literature reveals that training of HMMs can be done using the Baum-Welch algorithm, but to ensure proper and fast convergence of the models sensible initial values have to be calculated for the transitions parameters and the means and variances of the state density functions before the Baum-Welch algorithm can be used. HMMs are pretty sensitive to initial values so this stage is crucial for the entire process.

Good initial models can be obtained by assuming an HMM as a generator of speech vectors. The training examples of the phones corresponding to the model whose parameters are to be estimated can be viewed as the output of this model. Thus if the state that generated each vector in the training data was known, then the unknown means and variances could be estimated by averaging all the vectors associated with each state (Young et.al., 2002).

In HTK this concept is implemented using the HCompv tool. Based on the above prototype model, the tool HCompv was executed to create another average model.

hcompv -A -C config6 -f 0.01 -m -S mfclist.txt -M hmm0 proto

This produced a new prototype version and stored it in the directory hmm0. It scanned the set of training data files and compute the global mean and variance and set all of the Gausians in a given HMM to have the same mean and variance.

The –f option was used to create a variance floor macro, called vfloor, which is equal to 0.01 times the global variance and is used to set the floor on the variance estimated in the subsequent steps.

Finally, a Master Macro File (MMF) which contained the initial model set by the name hmmdefs was created. This was done by manually copying the prototype for each phone and renaming it accordingly.

4.2.3. Embedded re-estimation

Once the initial monophone model set was available, the next task was re-estimating the model parameters. And an embedded re-estimation strategy was used, that simultaneously updated all of the HMMs in the system using all of the training data.

In embedded training all models are trained in parallel rather than individually. That is each utterance is processed in turn and the accompanying transcriptions are used to construct a composite HMM which spans the whole utterance. This composite HMM is made and implemented by concatenating instances of the phone HMMs corresponding to

each label in the transcription. The Forward-Backward algorithm is then applied and the sums needed to form the weighted averages are accumulated. When all of the training files have been processed, the new parameter estimates are formed from the weighted sums and the updated HMM set is created.

This embedded procedure is implemented in the HTK tool *HERest* which performs exactly one iteration of the algorithm each time it is ran. It was executed by writing the following command:

C:\> herest -C config6 -I phones1.mlf -t 250.0 150.0 1500.0 -S mfclist.txt -H hmm0/macros -H hmm0/hmmdefs.mmf -M hmm1 monophones1.txt

Next the algorithm was run two more times for the models to get well adjusted better trained.

4.2.4. Fixing the silence models

In the previous step a 3 state left-to-right HMM was generated for each phone and for the silence model, sil. Though the sil model had the same topology as the other phones, it is supposed to take care of periods of silences that can vary greatly in length, from a few milliseconds up to a few seconds. Thus for this model to handle such situations, transitions had to be added from its second state to the fourth and from the fourth state back to the second. This was done to make the model more robust by allowing individual states to absorb the various impulsive noises in the training data. The back ward skip allows this to happen with out committing the model to transit to the following word.

So far, training was done with out the short pause model sp, because this optional silence could be misassigned to other models during initial trainings. But from this on, it was introduced using monophones2, the phone set with sp, which was generated using the tool HDman and the phone level MLF, phones2. This MLF was made to contain sp at the end of each word while it was generated using hled; out of a reasonable assumption that there will always be such unnoticed pause there in. Other wise, the transcription of each utterance had to be prepared in such a way that the sp was transcribed whenever it is observed in all the recorded speeches.

The model was designed to have three states, i.e., only one emitting state in the middle and two non-emitting states at left and right. Then the emitting state was made to be tied to the center state of the sil model resulting in the so called tee model, as shown in the following figure.

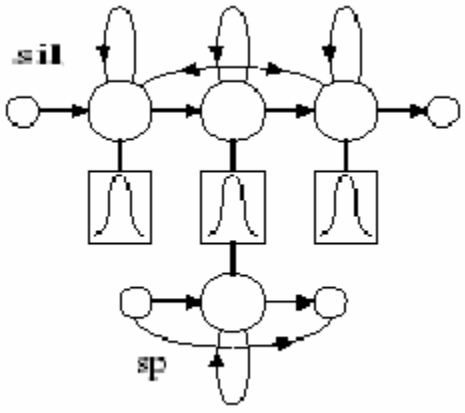

Fig 4.1. The silence model tied with the sp model.

The above two tasks were preformed using the HHed tool. First, the center state of the sil model was copied using a text editor, in order to make a new sp model. Accordingly, the transition matrix and the state numbers have been amended. Next all the models were put in a directory and the hled tool was run as follows using the required script:

HHEd –H hmm4/macros –H hmm4/hmmdefs –M hmm5 sil.hed monesphones2

The commands in the sil script add the transitions and then create the T model. Finally two more passes of HERest was made on these final HMMs resulting in the final monopphone based recognizer.

4.3. Refinements and Optimization

Literature indicate that monophone systems have a number of short comings and usually came up with poor performance. This basically is attributed to the fact that they are assumed to be insensitive to context variations. Thus they should be refined and strengthened. Accordingly, two of the commonest refining and optimization techniques were attempted in this experiment: mixture component splitting and promoting the monophones to tied state triphones. The activities undertaken along these lines and the results obtained are discussed below.

4.3.1. Multiple Mixtures

Multiple mixture systems are said to improve recognition results considerably, because they help avoid the problem resulting from the usage of the same type of distribution for different models and different states. If an HMM state is made to contain multiple mixture components, then the training vectors would be associated with highest likelihood mixture component. The number of vectors associated with each component within a state can then be used to estimate the mixture weights.

However incrementing the mixture size beyond certain limit has got one big disadvantage that it may result in over fitting i.e. as the number of mixtures increase, the models would fit more to the training data; in the end it results in models over fitted to the training data and generalizes poorly to other data.

During mixture incrementing some component weights may become very small, resulting in defunct mixture components. Defunct mixtures often indicate that not enough training data is available to further increase the mixtures of a model. So the best strategy employed in many systems is incrementing the mixture components in stages by a factor of N.

Thus taking the single Gausian monophone system above, the mixtures were incremented by a factor of two until 8 mxiture component HMMs were obtained. Then at each stage re-estimating and checking recognition results was performed.

In HTk, the conversion from single Gaussian HMMs to multiple mixture component HMMs is implemented using the HHEd MU command which will increase the number of components in a mixture by a process called mixture splitting. In this method the command works by repeatedly splitting the mixture with the largest mixture weight until the required number of components is obtained. This also allows recognition performance to be monitored to find the optimum.

By way of example, one of the scripts used, containing all the MU commands, is attached in appendix E.

4.3.2. Tied State triphones

The phone models described so far were context independent. There was a big assumption that the phonemes are more or less the same in every situation In reality this is not the case. As indicated in chapter two neighboring phonemes do influence each other in Amharic.

To capture these effects, called coarticulations, models are needed that take into account the context of a phone. One way of modeling coarticulation effects is using triphones. Triphones model the context by taking in to consideration the left and right neighboring phones. If two phones have the same identity but different left or right context they are considered as different triphones.

The triphone models constructed here were word internal because the other type, cross

word triphone, require far more data though they are powerful. Using HTK, the context-dependent triphones were prepared by simply cloning monophones and then re-estimating using triphones transcriptions.

First triphones' transcription was needed to train the triphone system. This was prepared using the HLEd command:

*HLEd –n triphones1 –l * -i wintri.mlf mktri.led phones1.mlf*

This will convert the mononophone transcriptions in phones1.mlf to an equivalent set of triphone level transcriptions in to wintry.mlf. At the same time, a list of triphones was written to the file triphones1. The edit script mktri.led contained the following commands:

```
WB sp
WB sil
TC
```

The WB commands defined sp and sil as word boundary symbols i.e. the start and end phones in a word.

The cloning of models was then performed by executing the following HHEd command:

HHEd –B –H hmm7/macros –H hmm7/hmmdefs.mmf –M hmm8 mktri.hed monophones1

The file mktri.hed was generated using a perl script, the source code of which is included in HTK Tutorial. The generated script is appended at appendix F. Then, two more re-estimation was performed, putting the model set in hmm10 directory.

The next task was making the tied state triphones. One of the mechanisms provided by HHEd was employed for this purpose. It was performed by running HHEd:

HHEd –B –H hmm10/macros –H hmm10/hmmdefs –M hmm11 tree.hed triphones1

The edit script tree.hed containing the instructions regarding which contexts to examine for possible clustering, was generated using the perl code provided in the HTK tutorial. A fragment of this script is attached at appendix G.

So far the set of triphones used, did cover only the training data. But here a new list of triphones was required, containing all triphones from all type of data. It was then created using the following command:

HDMan –b sp –n fulllist –g global.ded –l dicttri dictionsp

This command generated the new list called *fulllist* that is expanded to include all the triphones needed for recognition – both in the test and training data. Basically the new list was required by the AU command in the tree.hed script. This command then used the tree to synthesize all of the new previously unseen triphones in the new list[1].

Finally, and for the last time, the models are reestimated two twice using HERest.

[1] For descriptions and discussions of the other commands in the tree.hed, consult the HTK book.

4.4 Testing and Evaluation

Evaluation and testing were the final tasks in the process of developing a speech recognizer. In this study the progresses made at various levels in improving performance was measured using the development test data set. And in the end the performance of the developed recognizer was tested using the evaluation test set.

To check performances of resulting systems at all such levels, the viterbi decoding algorithm was used. This algorithm is implemented by the tool HVite in HTK. The transcriptions output by the viterbi algorithm were compared to the original word level transcription files using the HResult analysis tools. This tool uses a dynamic programming-based string alignment procedure.

The analysis tool computes the percentage of words correctly recognized using the following formula:

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

Where H stands for the number of labels recognized correct and N is the total number of labels in the test set.

In addition, the tool also performs word accuracy measure which takes into account the fact that some of the words classified as correct may be in fact insertion errors. This is

computed by:

$$Accuracy = \frac{H - I}{N} \times 100\%$$

Thus using this tool consecutive progress measures were done. There were important stages at which changes were expected and improvements were measured through out the process. Discussion of the results obtained and the accompanying explanations at those stages follows here.

The first evaluation was conducted on the model set resulted before the silence models were fixed. This is immediately after the first two consecutive embedded re-estimations i.e. on the model set stored at hmm3. Next testing was performed on the model set that resulted in to the monophone system. This is a recognizer solely based on monophnes i.e. the final model set before any refinement task was performed.

In this test and in all other tests conducted during development and refinement below, the development test set was used – 270 utterances spoken by 15 people. The results obtained at these two levels are compared and summarized by the following table.

System	% Word Correct	%Word Accuracy	%Sent. Correct
HMM set 3 (before the silence models)	15.74	-0.48	0.61
Monophone system (HMM7)	52.45	45.42	5.1

Table 4.2. Recognition results up to the monophone system.

The table displays that a big improvement on recognition was achieved after the models get more trained and the short pause and the silence models were added. The monophone system recognized three times as much words as the model set before silence models were added. The word accuracy was even negative and grew up to a 45.42 accuracy level.

Next after each refinement of the models by incrementing the component mixtures from single Gaussian to an 8 mixture components using the development test sentences. This was done simultaneously on three different PCs. The results obtained are also compared with the monophone (single Gaussian) system and presented as follows:

Number of Mixtures	% correct	% Accurate	%Correct Sentences
Single Gaussian	52.45	45.42	5.91
2	54.06	46.53	7.27
4	54.06	46.58	6.67
8	54.11	46.64	7.27

Table 4.3. Performance of model sets (recognizers) when component mixtures are incremented

This indicates that performance improvement is achieved from the single Gaussian to multiple mixtures. The increase, especially on the performances at the word level is linear – as the mixture is incremented the performance do also change, though the change is very small.

The other noticeable point is that the increment from 2 to 4 brought a decline in sentence level recognition performance, though there is a significant increase on it from the monophone system to the 2 mixture component. This very small discrepancy may be attributed to the small test dataset used.

The recognition tool, HVite was executed once again on the monophone models changing only two of its parameters - the word insertion penalty (-p)[1] and the grammar scale factors (-s)[2]. Young et. al (2002) indicated that these parameters can have a significant effect on recognition performance. The –p value was incremented from 0.2 earlier to 5.0 and the –s value was taken up to 10.0. The effect of these values is displayed by the following table.

System	% Word Correct	%Word Accuracy	%Sent. Correct
M(-p=0.2, -s=5.0)	52.45	45.42	5.91
M(-p=5.0, -s=10)	71.30	69.01	20.86

Table 4.4 – The monophone system with different –p and – s values.

The table shows a dramatic increase in performance when the two parameters are increased.

[1] The –p option is a fixed value added to each token when it transits from the end of one word to the start of the next.

[2] The grammar scale factor is the amount by which the language model probability is scaled before being added to each token as it transits from the end of one word to the start of the next.

Next, the triphone based system was examined on the development test data resulting in a big improvement.

%(words correctly recognized)	78%
%(words accurately recognized)	76.16%
%(sentences correctly recognized)	29%

Table 4.5. Triphones tested on the development data.

The table depicts a major change compared to the monophone systems in all the measurements. The improvements in both word and sentences are more interesting. The first set had a word level correctness of 15.74, but now it rose to 78%. The sentence level correctness went up to 29%from 0.6%. Especially the change on the word accuracy was more dramatic – from -0.48 to 76.16%.

However all these results were based on the development test set that they were compromised. They were used repeatedly. To test the real robustness of the system, the evaluation test data was used.

Thus, performance of the triphone system, the final recognizer, was measured using an evaluation test set. The following results were obtained.

%(words correctly recognized)	76.20
%(words accurately recognized)	74.97
%(sentences correctly recognized)	26.06

Table 4.4. Recognition results of the final triphone based system

The decline from the previous results was due to the very fact that the earlier triphone based system was tested with the development test data, that the models were being tested with it in all the development and the refinement stages.

Once again, this final system was tested after the mixture components of all its models were incremented to 8.

%(words correctly recognized)	79%
%(words accurately recognized)	76.18%
%(sentences correctly recognized)	30.01%

Table 4.5. Recognition result of the final with 8 mixture components

This test was done using the evaluation test set already used. Thus, the observed change might not be attributed wholly to the increment of the mixture component. Otherwise, this is the final result obtained by this experiment. The figures are not too close to the desired absolute systems. But given the circumstances, it is a highly encouraging result. With a little bit more work on optimization, better results can be achieved.

One final remaining test was performed for a data set recorded in a separate environment. Although the above evaluation data did not occur in the training or development test set, it was recorded under similar conditions – in the same room, same laptop computer, headset etc. Thus to cross check how the system lives up to a new data recorded on a different environment, it was tested with the data set made of 50 utterances by 5 people. The following result was observed:

%(words correctly recognized)	52%
%(words accurately recognized)	46.18%
%(sentences correctly recognized)	6.01%

Table 4.6. Recognition result of the final with 8 mixture components

Here the performance results clearly indicates major decline in comparison to the performance obtained in the last few results. These results were to be expected, since the data set was recorded in a different and very noisy environment. The data in the corpus, however, was recorded in a relatively silent room. The tools used here (like the microphone were also susceptible to interference).

Chapter 5

Conclusions and Recommendations

This chapter presents the conclusions drawn from the findings of the experiment and the researcher's recommendations on further actions that can be taken and future research areas.

5.1. Conclusions

Given the circumstances and the limitations, the result obtained can be said is promising. Above all things, the result of the experiment is a proof of the fact that it is possible to construct an Amharic speech recognizer using the HTK toolkit and the HMM modeling technique. Despite the low performance obtained, large vocabulary, speaker independent and continuously spoken Amharic speech recognizer can be built with further optimization efforts.

Literature indicate that phonemes are appropriate units of recognition for continuous and large vocabulary applications. But it is also discussed that they are context independent units that assume identical units where ever that unit is. Actually this is not true in reality. In languages like Amharic, a phone may display different features as the context varies.

This otherwise will have its own negative impact on recognizers performance. In this investigation, phonemes were taken as base units and to buttress up the mentioned side effect, they were promoted to a left-right context sensitive units, known as triphones. This has significantly boosted the performance from 71.30% word level accuracy to 76.20% word level accuracy in the triphone system; and from 20.86% to 26.06% sentence level accuracy.

The performance of such ASR is also known to be highly affected by the addition of mixture components to a certain level. In this experiment too a consistent increase was obtained. But the change was not that too big as expected.

The word insertion penalty (-p) and the grammar scale factors (-s) had a significant impact on the performance improvement of the recognizer.

5.2. Recommendations

This study was meant to focus on investigating the possibility of continuous, speaker independent, large vocabulary Amharic speech recognition. The natural extension of this work, along side with further optimization endeavors, is adopting the work to specific application areas that are discussed in section 1.5.

The modeling technique employed here in this study was pure HMM modeling. But there are other approaches currently being tried like hybriding HMMs and ANS (Aritficial Neural Networks). These newly emerging and plausible approaches are supposed to fill the gaps left out by pure HMMs. Thus further studies may investigate the possibilities of developing Amharic speech recognizers using those techniques or approaches.

It has been indicated that the recognizer demonstrated here was speaker independent but its independence is only to those people whose first language is Amharic or at least to those who can speak the language fluently. This is mainly because; the database was constructed from those native speakers. Thus, the study should be extended to be of service for all Tom, Dipty, Abebe and Tulu. This can be helped by using a corpus of much of much more data, developed from people with various socio-linguistic backgrounds.

It has been indicated in literature that one approach towards near speaker independence is speaker adaptation. Thus speaker adaptation may be attempted as a separate work by itself or as an extension of this work.

By way of stressing the point recommended in the previous works, the collaborative effort of people in the areas of linguistic, signal processing, computing and information science is highly important. This is demanded by the very nature of the field, especially if real applications are envisaged.

Buttressing up the above statements, one problem that faced the experiment is absence of pronunciation dictionary in the language. Thus construction of one such dictionary might be of much service.

References

Bender, L. M. *et al.*. "The Ethiopian Writing System." *The Languages of Ethiopia*. Ed. Bender, M. *et al.* , London: Oxford University Press, 1976 . 120-130.

Center for Speech Technology Research (CSTR). Espresso I: Phonetically featured Syllables for Speech Recognition. Available at http://www.cstr.ed.ac.uk/projects/espresso/espresso_1.html . 1999.

Cowley, R, *et al.*. "The Amharic Language." *Languages in Ethiopia.* Ed. Bender, M. *et al.*, London: Oxford University Press, 1976 . 77-90.

Ganapathiraju, Aravind. 2002. Support Vector Machines For Speech Recognition. A Dissertation Submitted to the Faculty of Mississippi State University

Gu, Liang. 2001. High-Performance Automatic Speech Recognition via Enhanced Front-end Analysis and Acoustic Modeling. A Dissertation. University of California: Santa Barbara.

Hamaker, Jonathan. 2002. A Dissertation Proposal Submitted to the Faculty of Mississippi State University. Mississippi State, Mississippi.

Jelinek, F. "Self-Organized Language Modeling for Speech Recognition." Readings in Speech Recognition. Eds. Waibel, Alex and Kai-Fu Lee, California: Morgan, 1985 . 450-507.

Kinfe Taddesse. 2001. Sub-Word Based Amharic Word Recognition: An Experiment Using Hidden Markov Model (Hmm)

Lee, K-F. 1987. Automatic speech recognition:the development of the SPHINX system. London: Kluwer Academic.

Lee, K-F. 1990. "Context-Dependent Phonetic Hidden Markov Models for Speaker-Independent Continuous Speech Recognition." Readings in Speech Recognition. Eds. Waibel and K-F, Lee. 347-362.

Markowitz, J. A. Using Speech Recognition. Upper Saddle River, New Jersey: Prentice Hall, Inc., 1996 .

Nahm, E. and D. Slater. "Speech Recognition." *The Ultimate Multimedia Handbook.* Ed. Keyes, Jessica , New York: McGraw-Hill, 1997 .

Odell, Jullian James. 1995. The Use of Context in Large Vocabulary Speech Recognition. Dissertation Submitted to the University of Cambridge: Queens College.

Rabiner, L.R., and B-H, Juang. Fundamentals of Speech Recognition. Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1993 .

Reiderer, K. Large Vocabulary Continuous Speech Recognition. Helsinki University of Technology: Laboratory of Computational Engineering. 1999.

Rodman, R. D. Computer Speech Technology. Norwood: Artech House, Inc., 1999 .

Srivastava, Shi, vali. 2002. Fast Gaussian Evaluations In Large Vocabulary Continuous Speech Recognition. A Thesis Submitted to the Faculty of Mississippi State

University.

Solomon Berhanu. Isolated Amharic Consonant-Vowel (CV) Syllable Recognition: An experiment Using the Hidden Markov Model. Master's Thesis, Addis Ababa University, Addis Ababa, 2001 .

Wiggers Paskal. 2001. Hidden Markov Models for Automatic Speech Recognition and their Multimodal Applictions. Delft University of Technology:The Netherlands.

Young, SJ. 1994. The HTK Hidden Markov Model Toolkit: Design and Philosophy. Cambridge University: Cambridge. Available at sjy@eng.cam.ac.uk

Young, Steve. 1996. Large Vocabulary Continuous Speech Recognition: a Review. Cambridge University: Cambridge.

Young, Steve et al.. 2000. The HTK Book. Microsoft Corporation.

Zue, V and R. Cole. "Spoken Language Input: Overview." Survey of the State of the Art in Human Language Technology. Eds. Cole, R.A et al., Oregon Graduate Institute, 1995 .

ባዬ ይማም "ፊደል እንደገና" ፣ የኢትዮጵያ ቋንቋዎችና የሥነ ጽሑፍ መጽሔት፣ ቁጥር 7፣ (1 – 32) ። 1997

Appendix A. The Amharic Phonemes

(Modified from Solomon's work)

Amharic Symbol	Etop font	In this paper	Amharic Word Example	
ም	m	m	/mar/	'honey'
ት	t	t	/tEmari/	'student'
ን	n	n	/nIgus/	'king'
ል	l	l	/lIb/	'heart'
ብ	b	b	/bet/	'house'
ር	r	r	/kIrIkIr/	'debate'
ይ	y	y	/ayn/	'eye'
ው	w	w	/wIyIyIt/	'discussion'
ስ	s	s	/sEw/	'man'
ግ	g	g	/gEmEd/	'rope'
ድ	d	d	/dabbo/	'bread'
ክ	k	k	/kErEmela/	'candy'
ች	^c	C	/ ĉIgIr/	'problem'
ጥ	.t	T	/t'Ewat/	'morning'
ቅ	k'	q	/k'ElEm/	'paint'
ህ	h	h	/hasab/	'idea'
ፍ	f	f	/fIlagot/	'interest'
ጅ	^g	ǰ	/ǰEmErE/	'he started'
ኝ	~n	N	/tE ň a/	'he slept'
ሽ	^s	S	/ šErErit/	'spider'
ዕ	.c, .s	x	/s'Ehai/	'sun'
ጭ	^C	X	/č'ErEk'a/	'moon'
ፕ	p	p	/polis/	'polis'
ወኧ	uA	@	/g^wdEňa/	'friend'
ዥ	^z	Z	/žIwažIwe/	'swing'
ዝ	z	z	/zEmEd/	'relative'

Amharic Symbol	Etop font	In this paper	Amharic Word Example	
ጵ	.p	P	/p'ap'p'as/	'bishop'
ኧ	a	a	/ErE/	'exclamation'
ኡ	u	u	/udEt/	'circulation'
ኢ	i	i	/itIyop'iya /	'Ethiopia'
ኣ	A	A	/abat/	'father'
ኤ	E	E	/eli /	'tortoise'
እ	e	e	/Inat/	'mother'
ኦ	o	o	/oromo/	'oromo'

Appendix B. The Amharic Consonants

Manner of articulation	Voicing	Place of Articulation						
		Bilabial	Labio-dental	Alveolar	Palatal	Velar	Labio-Velar	Glottal
Stops	Voiced	ለ /b/		D /d/		ገ/g/	ጓ/g^w/	
	Voiceless	ፐ /p/		ተ /t/		ከ/k/	ኳ/k^w/	ዐ (?)
	Ejective	ጰ /p'/		ጠ /t'/		ቀ/k'/	ቋ/k'^{w}/	
Fricatives	Voiced		ቨ /v/	ዘ /z/	ዠ /ž/			
	Voiceless		ፍ /f/	ስ /s/	> /š/			ሀ/h/ፉ ሏ /h^w/
	Ejective			ጸ /s'/				
Affricates	Voiced				ጀ /j/			
	Voiceless				ቸ /č/			
	Ejective				ጭ/c'/			
Nasals		ም /m/		N /n/	ኘ /ñ/			
Liquids				ል /l/	ር /r/			
Glides		W /w/			ይ /y/			

Appendix C. The major library and their modules in HTK

Name	*Module Function*
HADAPT	Provides support for the various HTK adaptation tools
HAUDIO	Direct audio input
HDICT	Interface for dictionary files
HGRAF	Interactive graphics interface
HLABEL	Label file i/o
HLM	Interfacing language model files
HMATH	Additional math support
HMEM	Memory management
HMODEL	HMM definition and i/o
HNET	Interface for networks and lattices
HPARSE	Grammar support
HREC	Contains the main recognition processing functions
HSHELL	User input/output and interaction with the operating system
HSINP	Signal processing operations needed for speech analysis
HTRAIN	Contains support for the various training tools

Appendix D. The Prototype HMM

```
~o <VecSize> 39 <MFCC_0_D_A>
~h "proto"
<BeginHMM>
  <NumStates> 5
  <State> 2
     <Mean> 39
        0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
     <Variance> 39
        1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

  <State> 3
     <Mean> 39
        0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
     <Variance> 39
        1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

  <State> 4
     <Mean> 39
        0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
     <Variance> 39
        1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

  <TransP> 5
   0.0 1.0 0.0 0.0 0.0
   0.0 0.6 0.4 0.0 0.0
   0.0 0.0 0.6 0.4 0.0
   0.0 0.0 0.0 0.7 0.3
   0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

Appendix E. A sample script for creating multiple mixture components

```
MU 8 {*A*.state[2-4].mix}
MU 8 {*C*.state[2-4].mix}
MU 8 {*E*.state[2-4].mix}
MU 8 {*n*.state[2-4].mix}
MU 8 {*d*.state[2-4].mix}
MU 8 {*a*.state[2-4].mix}
MU 8 {*w*.state[2-4].mix}
MU 8 {*o*.state[2-4].mix}
MU 8 {*m*.state[2-4].mix}
MU 8 {*r*.state[2-4].mix}
MU 8 {*N*.state[2-4].mix}
MU 8 {*e*.state[2-4].mix}
MU 8 {*l*.state[2-4].mix}
MU 8 {*i*.state[2-4].mix}
MU 8 {*k*.state[2-4].mix}
MU 8 {*S*.state[2-4].mix}
MU 8 {*X*.state[2-4].mix}
MU 8 {*T*.state[2-4].mix}
MU 8 {*t*.state[2-4].mix}
MU 8 {*b*.state[2-4].mix}
MU 8 {*u*.state[2-4].mix}
MU 8 {*g*.state[2-4].mix}
MU 8 {*f*.state[2-4].mix}
MU 8 {*q*.state[2-4].mix}
MU 8 {*y*.state[2-4].mix}
MU 8 {*h*.state[2-4].mix}
MU 8 {*Z*.state[2-4].mix}
MU 8 {*s*.state[2-4].mix}
MU 8 {*z*.state[2-4].mix}
MU 8 {*j*.state[2-4].mix}
MU 8 {*x*.state[2-4].mix}
MU 8 {*p*.state[2-4].mix}
MU 8 {*v*.state[2-4].mix}
MU 8 {*P*.state[2-4].mix}
MU 8 {*sil*.state[2-4].mix}
```

Appendix F. The mktri.hed script that is used for cloning

```
CL triphones1.txt
TI T_A {(*-A+*,A+*,*-A).transP}
TI T_sp {(*-sp+*,sp+*,*-sp).transP}
TI T_d {(*-d+*,d+*,*-d).transP}
TI T_e {(*-e+*,e+*,*-e).transP}
TI T_n {(*-n+*,n+*,*-n).transP}
TI T_C {(*-C+*,C+*,*-C).transP}
TI T_a {(*-a+*,a+*,*-a).transP}
TI T_w {(*-w+*,w+*,*-w).transP}
TI T_o {(*-o+*,o+*,*-o).transP}
TI T_N {(*-N+*,N+*,*-N).transP}
TI T_k {(*-k+*,k+*,*-k).transP}
TI T_S {(*-S+*,S+*,*-S).transP}
TI T_X {(*-X+*,X+*,*-X).transP}
TI T_l {(*-l+*,l+*,*-l).transP}
TI T_T {(*-T+*,T+*,*-T).transP}
TI T_r {(*-r+*,r+*,*-r).transP}
TI T_t {(*-t+*,t+*,*-t).transP}
TI T_b {(*-b+*,b+*,*-b).transP}
TI T_i {(*-i+*,i+*,*-i).transP}
TI T_u {(*-u+*,u+*,*-u).transP}
TI T_g {(*-g+*,g+*,*-g).transP}
TI T_@ {(*-@+*,@+*,*-@).transP}
TI T_f {(*-f+*,f+*,*-f).transP}
TI T_m {(*-m+*,m+*,*-m).transP}
TI T_E {(*-E+*,E+*,*-E).transP}
TI T_q {(*-q+*,q+*,*-q).transP}
TI T_y {(*-y+*,y+*,*-y).transP}
TI T_h {(*-h+*,h+*,*-h).transP}
TI T_s {(*-s+*,s+*,*-s).transP}
TI T_j {(*-j+*,j+*,*-j).transP}
TI T_z {(*-z+*,z+*,*-z).transP}
TI T_Z {(*-Z+*,Z+*,*-Z).transP}
TI T_x {(*-x+*,x+*,*-x).transP}
TI T_p {(*-p+*,p+*,*-p).transP}
TI T_v {(*-v+*,v+*,*-v).transP}
TI T_P {(*-P+*,P+*,*-P).transP}
TI T_sil {(*-sil+*,sil+*,*-sil).transP}
```

Appendix G. Fragment of the tree.hed script

```
RO 100.0 stats
TR 0
QS "L_Stops" {b-*,d-*,g-*,p-*,t-*,k-*,P-*,T-*,*+q}
QS "R_Stops" {*+b,*+d,*+g,*+p,*+t,*+k,*+P,*+T,*+q}
QS "L_Fricatives" {v-*,z-*,Z-*,f-*,s-*,S-*,h-*,*+x}
QS "R_Fricatives" {*+v,*+z,*+Z,*+f,*+s,*+S,*+h,*+x}
QS "L_Affricates" {j-*,C-*,*+X}

.


QS "L_A" {A-*}
QS "R_A" {*+A}
QS "L_sp" {sp-*}
QS "R_sp" {*+sp}
QS "L_d" {d-*}
QS "R_d" {*+d}
QS "L_e" {e-*}
QS "R_e" {*+e}

.


TR 2

TB 350.0 "ST_A_2_" {(A,*-A+*,A+*,*-A").state[2]}
TB 350.0 "ST_sp_2_" {(sp,*-sp+*,sp+*,*-sp").state[2]}
TB 350.0 "ST_d_2_" {(d,*-d+*,d+*,*-d").state[2]}
TB 350.0 "ST_e_2_" {(e,*-e+*,e+*,*-e").state[2]}
TB 350.0 "ST_n_2_" {(n,*-n+*,n+*,*-n").state[2]}
TB 350.0 "ST_C_2_" {(C,*-C+*,C+*,*-C").state[2]}
TB 350.0 "ST_a_2_" {(a,*-a+*,a+*,*-a").state[2]}

.


TB 350.0 "ST_p_4_" {(p,*-p+*,p+*,*-p").state[4]}
TB 350.0 "ST_v_4_" {(v,*-v+*,v+*,*-v").state[4]}
TB 350.0 "ST_P_4_" {(P,*-P+*,P+*,*-P").state[4]}
TB 350.0 "ST_sil_4_" {(sil,*-sil+*,sil+*,*-sil").state[4]}

TR 1

AU "fultri"

CO "tiedlist"

ST "trees"
```

APPENDIX H. The configuration parameter used at coding

```
#coding parameters
SOURCEFORMAT = WAVE
TARGETFORMAT = HTK
TARGETKIND = MFCC_0_D_A
TARGETRATE = 100000.0
SAVECOMPRESSED = T
SAVEWITHCRC = T
WINDOWSIZE = 250000.0
USEHAMMING = T
PREEMCOEF = 0.97
NUMCHANS = 26
CEPLIFTER = 22
NUMCEPS = 12
ENORMALISE = F
```

Appendix I

Generated Lattice Format

```
N=28667 L=133506
I=0    W=!NULL
I=1    W=!ENTER
I=2    W=A
I=3    W=AAde
I=4    W=AAdene
I=5    W=ACA
I=6    W=ACACawe
I=7    W=ACAwe
I=8    W=ACAwene
I=9    W=ACAwoCACawe
I=10   W=ANewAke
I=11   W=ASASAXe
I=12   W=ASASAla
I=13   W=ASATere              I=14   W=ASAgAri
…

I=28666 W=!EXIT
J=0     S=1    E=0    l=-1.26
J=1     S=2    E=0    l=-0.69
J=2     S=3    E=0    l=-0.69
J=3     S=4    E=0    l=-0.98
J=4     S=5    E=0    l=-1.87
J=5     S=6    E=0    l=-0.69
J=6     S=7    E=0    l=-0.69
J=7     S=8    E=0    l=-0.69
J=8     S=9    E=0    l=-0.69
J=9     S=10   E=0    l=-0.69
J=10    S=11   E=0    l=-0.69
J=11    S=12   E=0    l=-0.60
J=12    S=13   E=0    l=-0.69
J=13    S=14   E=0    l=-0.69
…

J=133497 S=28507 E=28666 l=-0.29
J=133498 S=28520 E=28666 l=-0.06
J=133499 S=28523 E=28666 l=-0.03
J=133500 S=28527 E=28666 l=-0.69
J=133501 S=28537 E=28666 l=-0.29
J=133502 S=28538 E=28666 l=-0.69
J=133503 S=28579 E=28666 l=-0.69
J=133504 S=28650 E=28666 l=-0.69
J=133505 S=28660 E=28666 l=-0.69
```